A COHERENT VLSI DESIGN ENVIRONMENT

Semiannual Technical Report for the period October 1, 1986 to March 31, 1987

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Principal Investigators:		
	Paul Penfield, Jr.	(617) 253-2506
	William J. Dally	(617) 253-6043
	Lance A. Glasser	(617) 253-4677
	Thomas F. Knight, Jr.	(617) 253-7807
	F. Thomson Leighton	(617) 253-3662
	Charles E. Leiserson	(617) 253-5833
	John L. Wyatt, Jr.	(617) 253-6718

This research was sponsored by Defense Advanced Research Projects Agency (DoD), through the Office of Naval Research under ARPA Order No. 3872, Contract No. N00014-80-C-0622.

TABLE OF CONTENTS

Selected Publications (starting after page 12)

L. A. Glasser, "Frequency Limitations in Circuits Composed of Linear Devices," to appear as "Frequency Limitations in Linear Circuits" in Proceedings, 1987 IEEE International Symposium on Circuits and Systems, Philadelphia, PA, May 4-7, 1987. Also, MIT VLSI Memo No. 86-348, November 1986.

T. S. Hohol and L. A. Glasser, "RELIC: A Reliability Simulator for Integrated Circuits," Proceedings, International Conference on Computer-Aided Design, Santa Clara, CA, November 11-13, 1986. To appear (translated into Japanese) in Nikkei Microdevices. Also, MIT VLSI Memo No. 87-360, January 1987.

A. V. Goldberg and S. A. Plotkin "Efficient Parallel Algorithms for $(\Delta + 1)$-Coloring and Maximal Independent Set Problems," Laboratory for Computer Science, MIT, Technical Memorandum MIT-LCS-TM-320, January 1987.

T. H. Cormen, "Efficient Multichip Partial Concentrator Switches," Laboratory for Computer Science, MIT, Technical Memorandum MIT-LCS-TM-322, February 1987.

*A. V. Goldberg, Efficient Graph Algorithms for Sequential and Parallel Computers, Ph.D thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, February 1987. Also, Laboratory for Computer Science Technical Memorandum MIT-LCS-TR-374, February 1987.

*J. L. Wyatt, Jr., "Nonlinear Dynamic Maximum Power Theorem," Research Laboratory of Electronics, MIT, RLE Technical Report. No. 525, March, 1987.

---

* Abstract only. Complete version available from Microsystems Research Center, Room 39-321, MIT, Cambridge, MA 02139; telephone (617) 253-8138.

RESEARCH OVERVIEW

It has been an open question in electrical network theory whether it is possible to synthesize a network with a prescribed natural frequency (in the complex s-plane) out of a restricted class of components. It was possible to say since the 1950s that no natural frequencies can be obtained in certain parts of the s-plane, but not the converse, namely that at other points a circuit can be devised. This question is of modern importance since the components in question may be MOS transistors and RC lines, whose accurate high-frequency models are complicated. It is important to know how fast a rise time can be achieved, or at what frequencies unwanted oscillations might occur. A set of necessary and sufficient bounds are now possible, in the sense that every point in the s-plane can be easily discovered to be either a frequency at which oscillation cannot occur with any possible combination of components and ideal transformers, or else a point at which a circuit consisting of a few such components and ideal transformers can be made to oscillate. The inclusion of ideal transformers is necessary since otherwise generally only a finite or countable number of natural frequencies can be found.

Many new results have been derived for parallel algorithms and complexity, such as one of the most astonishing is that a hypercube with a large number of faulty nodes can be used, with high probability, as another perfectly functioning hypercube of half the size, by using reconfiguration algorithms that are simple, fast, and require only local information.

The design of a message-driven processor continues. It is now being realized that many different highly parallel architectures require roughly the same sort of processing node, one that can respond quickly (i.e., with low latency) to messages that may require execution of a few (say about ten) instructions. The processor being designed can be considered an experiment in unifying message-passing and shared-memory architectures.

The waveform bounding work is continuing at a slightly slower pace because Prof. John Wyatt is at Caltech working with Carver Mead for the spring semester. Nevertheless there is progress in two areas. New macromodels have been developed for ECL circuits, and a depth-interpolation vision chip has been designed and submitted for fabrication.

The work on Schema is winding down, as students working directly on Schema projects finish up. Prof. Richard Zippel, who was responsible for Schema, has now left MIT. This project was very effective as a bridge among several research faculty and students, and has in that sense served its purpose.

Two of the projects supported by this contract were reported at the recent Conference on Advanced Research in VLSI, held at Stanford University, March 23-25, 1987. These are an analysis of multiprocessor communication networks, and a technique for powering and communicating with an IC chip without having any physical connections such as wires.

THE WAVEFORM BOUNDING APPROACH TO TIMING ANALYSIS

The timing analyzer for ECL standard cell design, mentioned in the last report, approaches completion. It will definitely be used as an in-house design tool by DEC if it works as we expect.

Gates in the ECL library are represented by macromodels. The macromodel parameters must be chosen carefully so that the macromodel waveforms closely resemble the output of computation-intensive SPICE simulations of the gates when loaded by fanout to other gates through interconnect. The straightforward approach involves a lot of least-squares curve fitting with iterative variation of the macromodel parameters and is therefore also very computation-intensive. In response to this problem, Peter O'Brien has cleverly reformulated the macromodel so that the first two moments of the macromodel waveforms can be calculated in advance in closed form. These explicit formulas are then used in the parameter-fitting process, resulting in a dramatic savings in computer time. It is possible that this type of macromodel will become more widely used because of this advantage.

We have submitted CIF files to MOSIS for our other project, the design of an analog depth-interpolation chip for computer vision applications. This first design was more of a technology-exploration effort than an attempt to produce a useful product. The idea is to use a regular planar array of linear resistors to rapidly interpolate a 2-D array of voltages that encode a (noisy and incomplete) array of depth estimates for a 3-D scene. A fundamental problem in the initial approach is that any such linear interpolation scheme not only smooths out noise and missing data, it also smooths over depth boundaries where actual discontinuities occur in the scene. Through consultation with Prof. Carver Mead at Caltech, we have found a way to synthesize a nonlinear "resistor" with a characteristic of the form $i = I \times \tanh(qv/kT)$. Such a current-limited "resistor" appears linear for $|v| << kT/q$, but supplies a maximum current of magnitude I regardless of the potential drop across it. In our application it should smoothly interpolate small changes in depth, but provide clean "breaks" at depth discontinuities.

HIGH PERFORMANCE CIRCUIT DESIGN

Maximum Frequency of Oscillation: We have now shown that the bounds on the maximum frequency of oscillation of linear networks discussed in the last progress report are tight. That is, not only can we say when a set of components cannot produce a circuit which can oscillate, but we can now say that unless we predict that it cannot oscillate, then a circuit can be found that does oscillate. We have also developed a program on the Symbolics LISP machine that will evaluate complex component models to see whether circuits built of them can or cannot oscillate. This program has been tested on a transistor small-signal model with about two dozen elements. A C version is under development.

Modeling of magnetostatic couplers for CMOS VLSI chips: We have now demonstrated inductive power coupling into a zero-pin chip. By the use of an on-chip bridge rectifier, voltages up to 10 V dc and powers up to 1 mW have been successfully coupled into a bulk CMOS chip using an HP 3312A function generator driving an external coil.

ARCHITECTURAL DESIGN

Prof. Leighton is continuing work on wafer-scale integration of systolic arrays, parallel algorithms and architectures, and fault-tolerance. In the area of wafer-scale integration, he and a student (John Burroughs) are developing efficient algorithms for integrating 2-dimensional arrays on a wafer containing randomly located faults. The current work extends the theoretical work reported in the May, 1986 VLSI research review by developing and coding algorithms for cell assignment and routing that work well experimentally for arrays of sizes 5×5 to 100×100. This work will be described in John Burrough's Bachelor's Thesis and a forthcoming technical report. It is a nice example of good theoretical ideas and asymptotic proofs being adapted to work experimentally on realistic problems.

In the area of parallel algorithms and architectures, Prof. Leighton and Bruce Maggs are developing efficient hash functions for many-one routing on a hypercube. Although greedy algorithms have long been known to work well for one-one routing problems on the hypercube, many-one routing problems can be much more difficult. In fact, if the destinations of the requests are not randomized, many-one routing problems can be intractable even if combining is allowed. In the present work, Leighton and Maggs are developing simple hash functions for randomizing shared memory locations so that any many-one routing problem can be efficiently solved by a greedy type of algorithm. Many-one routing problems on the hypercube are important since they provide the basis for simulating a CRCW PRAM, a very general and powerful architecture-independent model for parallel computation. One-one routing problems, on the other hand, are only sufficient to simulate an EREW PRAM, a much weaker model of parallel computation. The difficulty of many-one routing problems has been observed in many contexts including the speed with which routing can be performed on a Connection Machine. It is hoped that the present work will lead to substantially improved routing algorithms.

Also in the area of parallel algorithms and architectures, Prof. Leighton and non-MIT coworkers have discovered very efficient ways of using the hypercube to simulate special purpose architectures, without paying the usual overhead due to routing. For example, it has long been known that an N-node hypercube contains any N-node array of any dimension as a subgraph. Hence array-based algorithms can be (and frequently are) directly implemented on a hypercube without any overhead. Recently, Leighton et. al. have shown that any binary tree can also be embedded in the hypercube. Binary tree structures are useful in some numerical and parsing calculations as well as in the implementation of some divide-and-conquer algorithms. In addition, Prof. Leighton has shown that the powerful mesh of trees network is also a subgraph of the hypercube and hence a large number of graph and matrix calculations can be directly implemented on the hypercube. For example, the transitive closure of an N-node graph can now be computed on an N^2-node hypercube in $O(\log^2 N)$ steps, and two $N \times N$ matrices can be multiplied in $O(\log N)$ steps on an N^3-node hypercube. Although not difficult, the embedding of the mesh of trees in the hypercube is nonobvious, and it dramatically increases the ability of the hypercube to perform special purpose calculations.

In the area of fault-tolerance, John Hastad, Prof. Leighton, and Mark Newman are developing ways of reconfiguring a hypercube in the presence of a potentially large number of randomly located faults. Thus far, the work has been very promising. Among other things, Hastad, Leighton and Newman have shown that with high probability, an N/2-node hypercube can be one-one embedded into the live nodes of an N-node hypercube containing pN randomly located faults ($p < \frac{1}{2}$) so that neighboring nodes of the N/2-node hypercube are mapped to nodes at distance 3 or less apart in the N-node hypercube. Hence a hypercube containing a very large number of randomly located faults has virtually the same computational power as a fully functioning hypercube! Moreover, the algorithm for reconfiguring the hypercube is simple, fast, and works using only local control.

James K. Park has been working on a deterministic, on-line message-routing algorithm for a variant of the fat-tree that uses only constant-sized switches. This algorithm routes arbitrary message sets M in $O(\lambda(M) \log^2 n)$ bit steps. The algorithm is interesting in that it uses the network both for routing messages and for computing how much congestion there is in various parts of the network. Charles Leiserson and James Park are currently revising and simplifying the algorithm.

Tom Cormen continued his work on concentrator switch designs, showing that a switch that ε-nearsorts its incoming valid bits is an $(n, m, 1 - \varepsilon/m)$ partial concentrator switch. Using this result, he designed multichip partial concentrator switches based on two algorithms for sorting on a mesh. The first, based on Revsort (Schnorr and Shamir), is an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator with at most $2\sqrt{n} + \lfloor (\lg n)/2 \rfloor$ data pins per chip, $\Theta(\sqrt{n})$ chips, and volume $\Theta(n^{3/2})$ in which a message incurs $3 \lg n + O(1)$ gate delays. The second switch, based on Columnsort (Leighton), is an $(n, m, 1 - O(n^{2-2\beta}))$ partial concentrator switch with $\Theta(n^{\beta})$ data pins per chip, $\Theta(n^{1-\beta})$ chips, and volume $\Theta(n^{1+\beta})$, for any $\frac{1}{2} \leq \beta \leq 1$. A message incurs $4\beta \lg n + O(1)$ gate delays in passing through this switch.

Cynthia Phillips and Charles Leiserson completed their development of a simple parallel algorithm for contraction of n-node bounded-degree planar graphs which solves the problem of region labeling in vision systems and leads to algorithms to compute spanning trees and biconnected components of bounded-degree planar graphs. The connected components algorithm runs in $O(\lg n)$ randomized time on the restrictive exclusive-read exclusive-write PRAM model. They are currently trying to extend these results to planar graphs of arbitrary degree. Phillips is trying to develop an $O(\lg n)$-time parallel algorithm for region labeling in n-voxel 3D images by exploiting the restrictive topology and geometry of the spatial tesselation. She is also investigating the use of randomized search techniques for fast, simple connected components algorithms for general graphs.

Shlomo Kipnis, Joe Kilian, and Charles Leiserson have been investigating the power of bussed interconnection schemes for realizing permutations among chips. They established a correspondence between bussed permutation architectures and difference covers for permutation sets. As an example, only $\sqrt{n}$

pins per chip are needed to realize all cyclic shifts among n chips in one clock cycle. Using point-to-point wires, $n - 1$ pins per chip are required. They also show that $O(\sqrt{n})$ pins per chip suffice to realize any abelian group of permutations, and any general group of permutations requires $O(\sqrt{n \lg n})$ pins per chip. Some other results include bussed interconnection schemes for hypercubes ($d + 1$ pins per chip), shuffle-exchange graphs (3 pins per chip), and d-dimensional meshes ($d + 1$ pins per chip).

Alex Ishii and Charles Leiserson have continued work on understanding the timing of level-clocked synchronous circuits. For periodic clocking disciplines, they now have an efficient graph-theoretic algorithm for determining whether a circuit operates properly.

Miller Maley is completing his Ph.D. on planar routing. He has provided a solid topological framework for understanding routing problems in geometric terms, which has led to good algorithms for routing, routability testing, and compaction with automatic jog introduction.

Andrew Goldberg has completed his Ph.D. on parallel and sequential algorithms for graph problems. Among his recent results is a new algorithm (joint with R. E. Tarjan of Princeton) for determining the minimum-cost flow in a network. The algorithm is based on a new successive-approximation technique and is theoretically the best algorithm to date for many instances of the problem. Applications of mincost flow in VLSI include pad routing and optimally terminating interconnections. Its applications in the field of operations management are more extensive.

Paul Beame, Tom Leighton, and Charles Leiserson observed that a shuffle-exchange graph or a butterfly network on n nodes can simulate an n-node hypercube with a slowdown of only $O(\lg \lg n)$, instead of the usual $O(\lg n)$, if the simulation is off-line and the hypercube uses only one dimension at a time.

Serge Plotkin observed that the firing-squad problem, normally stated for a linear array of finite automata, could be solved on an arbitrary bounded-degree graph in time proportional to the diameter of the graph.

Andrew Goldberg and Serge Plotkin have been developing efficient parallel algorithms for symmetry-breaking in sparse graphs of processors. The symmetry-breaking algorithms give efficient ways to convert probabilistic algorithms to deterministic algorithms. Some of the techniques have been applied to construct several efficient linear-processor algorithms for graph problems, including an $O(\lg^* n)$-time algorithm for $(\Delta + 1)$-coloring of constant-degree graphs.

Philip Klein has developed, in collaboration with John Reif of Duke, an efficient parallel algorithm for planarity. On n-node graphs, the algorithm works in $O(\log^2 n)$ time using only n processors, in contrast to the previous best algorithm which used about n^3 processors to achieve the same time bound. In October, 1986, he presented the research at the IEEE Symposium for Foundations of Computer Science. He is currently working on representa-

tions of directed graphs that permit reachability queries to be answered efficiently in parallel.

Paul Beame has been working on methods for proving lower bounds on the resources needed by parallel computers in order to solve various computational problems. He has extended the results of his previous joint work with Johan Hastad in this area to include new graph-theoretic problems including finding small cliques in graphs. He is also investigating the computational advantages of concurrent-read memory access for parallel computers. Paul has improved the running time of the best known parallel machine algorithms for symmetry-breaking on bounded-degree graphs to $O(\log \log^* n)$. Also, with Baruch Awerbuch, he has shown that for distributed computation there are several infinite families of graphs for which known symmetry-breaking algorithms are optimal.

SYSTEMS

Message-Driven Processor

The natural grain size of many parallel algorithms is about 10 instructions. To fully exploit the concurrency in such algorithms, we must be able to efficiently execute tasks of this length. The message transmission and reception overhead of existing systems is in excess of 200 instruction times. With such a large overhead, these systems must execute tasks at the artificially large grain size of about 1,000 instructions. If we can reduce the overhead and operate at the natural grain size, we can effectively apply 100 times as many processing elements to the problem.

The Message-Driven Processor (MDP) is a processing element for a fine-grain, message-passing concurrent computer. This 36-bit, tagged machine is being designed as a single-chip processing element incorporating on-chip memory and router. We are using the MDP as a vehicle for experimenting with methods of reducing processor latency on message reception. The current version of the MDP can respond to an arriving message in less than 0.5 μs as compared to the 300 μs response time in the Intel iPSC. This low latency is achieved by providing hardware for queuing messages and for dispatching instruction sequences in response to message arrival.

The control mechanism of the MDP is driven by the arriving message stream. The MDP dispatches control on the basis of messages arriving over the network just as a conventional processor dispatches control on the basis of instructions fetched from memory. When a message is received by an MDP, a decision is made in hardware to either queue the message (without slowing the executing task), or to interrupt current task and execute the message. If the message is to be executed, control is immediately dispatched to the appropriate instruction sequence and useful instructions are executed five clock cycles after message arrival.

To achieve a fast context switch, the MDP is a memory-based rather than register-based machine. Only four user registers need be saved on a task switch, and two register sets are provided so that the most common task switches can be performed with no saving. The MDP can access its on-chip memory in a single clock cycle, eliminating the need for a large register set. Each MDP instruction may take one operand from memory. A small register set is used to provide the remaining two operands since the cost of multi-porting the memory is excessive. Many of the techniques used on uniprocessors to keep data on chip (load/store instruction sets, register windows, stack caches, etc.) do not work well on a multicomputer where context switches happen every 10 instructions as opposed to every 25,000, where there is little LIFO allocation of stack frames. With the MDP we have found that a simple memory-based instruction set operating out of a small on-chip memory supports the needs of multicomputer programs.

The MDP is also an experiment in unifying shared-memory and message-passing parallel computers. Shared-memory machines provide a uniform global name

space (address space) that allows processing elements to access data regardless of its location. Message-passing machines perform communication and synchronization via node-to-node messages. These two concepts are not mutually exclusive. The MDP provides a virtual addressing mechanism intended to support a global name space while using an execution mechanism based on message passing. While our plans are to program the machine using an actor model of computation we provide mechanisms to efficiently support models ranging from dataflow to communicating sequential processes.

The MDP uses a memory architecture that allows both indexed and set-associative accesses to its on-chip memory. The set-associative mode is used by the run-time system to provide virtual addressing, and to accelerate late-binding method lookup. By building comparators into the column multiplexer of the on-chip RAM, we are able to provide set-associative access with only a small increase in the size of the RAM's peripheral circuitry. The MDP is the only machine we are aware of that makes the address translation mechanism explicitly available to the programmer. In writing system code, we have found it to be an extremely useful feature.

Over the past six months we have constructed three register-transfer simulators of the MDP. Each simulator marked a step in the evolution of the MDP to its present form. A major step was the elimination of a large hard-wired message set in favor of a single message type (execute) and the use of instruction sequences to define more complex messages. Our simulations showed that defining messages with instruction sequences required only one or two additional clock cycles over the hard-wired approach. The added flexibility and the simplification of the design resulting from a single message type more than offset this small performance penalty. The flexibility is particularly important in an experimental machine. For example, we can redefine system messages such as 'SEND' or 'CALL' to add instrumentation. The addressing mechanisms used to access instruction streams and message arguments have also evolved through simulation and by coding many of the run-time system routines. Layout studies for critical components of the MDP are currently under way.

Bidirectional Torus Router

The Bidirectional Torus Router (BTR) is a self-timed, multicomputer communication chip that we are using to experiment with techniques for improving the performance of multicomputer communication networks.

One idea we are testing with the BTR is using virtual channels to multiplex two logical communication networks on a single physical network. The BTR provides two classes of communication, user and system, that have separate buffers but share the same physical wires. The two networks are logically completely separate. Our system software uses this separation to provide priority service for critical messages. Also, if the user network becomes congested (perhaps due to queue overflow in a processing node) we transmit emergency messages on the system network to clear the problem.

We also plan to use the BTR to evaluate the use of:

1. A Galois counter to reduce routing delay. The Galois counter can decrement in a single gate delay as opposed to about 8 gate delays for a previous chip (TRC) using an integer counter.

2. A single wire request/acknowledge line to reduce pin count.

3. A distributed token-passing arbiter. This arbiter allows us to efficiently multiplex two directions of communication on a single channel. As long as one chip holds the token, it can use the channel without paying a round-trip delay to the other chip for arbitration.

4. A partitioned data path rather than a crossbar switch to reduce router area.

During the past six months we have completed a revision of the BTR logic design and have begun simulations to verify the design.

Performance of k-ary n-cube Interconnection Networks

We have studied the behavior of k-ary n-cube interconnection networks under varying traffic using both simulation and queuing models. We have developed an analytical model of latency as a function of offered traffic in unbuffered k-ary n-cube interconnection networks that agrees with network simulation results to within 5%. Both simulation and analysis indicate that latency grows slowly with offered traffic until a saturation point of about 50% capacity. This result implies that the latency advantage of low-dimensional k-ary n-cubes is not degraded by traffic as long as the network is operated below the saturation point.

PUBLICATIONS LIST

S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg, "Optimal Simulations of Tree Machines," Proc. 27th IEEE Conference on Foundations of Computer Science, Portland, OR, October, 1986, pp. 274-282. Also, MIT VLSI Memo No. 86-354, December 1986.

R. E. Zippel, P. Penfield, Jr., L. A. Glasser, C. E. Leiserson, J. L. Wyatt, Jr., F. T. Leighton, and J. Allen, "Recent Results in VLSI CAD at MIT," Proc. 1986 Fall Joint Computer Conference, Dallas, TX, November 2-4, 1986, pp. 871-877. Also, MIT VLSI Memo No. 86-341, October 1986.

T. S. Hohol and L. A. Glasser, "RELIC: A Reliability Simulator for Integrated Circuits," Proceedings, International Conference on Computer-Aided Design, Santa Clara, CA, November 11-13, 1986. To appear (translated into Japanese) in Nikkei Microdevices. Also, MIT VLSI Memo No. 87-360, January 1987.

F. Chung, T. Leighton, and A. Rosenberg, "Embedding Graphs in Books: A Layout Problem with Applications to VLSI Design," SIAM J. Algebraic and Discrete Methods, vol. 8, no. 1, Jan. 1987, pp. 33-58. Also, MIT VLSI Memo No. 85-235, March 1985.

A. V. Goldberg, Efficient Graph Algorithms for Sequential and Parallel Computers, Ph.D thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, February 1987. Also, Laboratory for Computer Science Technical Memorandum MIT-LCS-TR-374, February 1987.

C. W. Selvidge, A. C. Malamy, and L. A. Glasser, "Power and Communication Techniques for Physically Isolated Integrated Circuits," Proceedings, 1987 Conference on Advanced Research in VLSI, Stanford, CA, March 23-25, 1987, P. Losleben, ed., MIT Press, Cambridge, MA, 1987, pp. 231-247.

W. J. Dally, "Wire-Efficient VLSI Multiprocessor Communication Networks," Proceedings, 1987 Conference on Advanced Research in VLSI, Stanford, CA, March 23-25, 1987, P. Losleben, ed., MIT Press, Cambridge, MA, 1987, pp. 391-415.

L. A. Glasser, "Frequency Limitations in Circuits Composed of Linear Devices," to appear as "Frequency Limitations in Linear Circuits" in Proceedings, 1987 IEEE International Symposium on Circuits and Systems, Philadelphia, PA, May 4-7, 1987. Also, MIT VLSI Memo No. 86-348, November 1986.

J. Hastad, T. Leighton, and M. Newman, "Reconfiguring a Hypercube in the Presence of Faults," to appear in Proc. 1987 ACM Symposium on Theory of Computation.

A. V. Goldberg and S. A. Plotkin,"Parallel $(\Delta + 1)$ Coloring of Constant-Degree Graphs," to appear in Information Processing Letters.

INTERNAL MEMORANDA

A. V. Goldberg and S. A. Plotkin "Efficient Parallel Algorithms for $(\Delta + 1)$-Coloring and Maximal Independent Set Problems," Laboratory for Computer Science, MIT, Technical Memorandum MIT-LCS-TM-320, January 1987.

T. H. Cormen, "Efficient Multichip Partial Concentrator Switches," Laboratory for Computer Science, MIT, Technical Memorandum MIT-LCS-TM-322, February 1987.

J. L. Wyatt, Jr., "Nonlinear Dynamic Maximum Power Theorem," Research Laboratory of Electronics, MIT, RLE Technical Report. No. 525, March, 1987.

TALKS WITHOUT PROCEEDINGS

A. V. Goldberg, "A New Approach to the Maximum Flow Problem," MIT VLSI Research Review, Cambridge, MA, December 15, 1986.

J. Hastad, T. Leighton, and M. Newman, "Fault Tolerance in Hypercubes," MIT VLSI Research Review, Cambridge, MA, December 15, 1987.

J. Kilian, S. Kipnis, and C. E. Leiserson, "The Organization of Permutation Architectures with Multiple-Pin Interconnections," MIT VLSI Research Review, Cambridge, MA, December 15, 1986.

C. Selvidge and A. Malamy, "Magnetostatic I/O Techniques for Integrated Circuits," MIT VLSI Research Review, Cambridge, MA, December 15, 1986.

J. L. Wyatt, Jr., "Recent Progress on Delay Bounds for MOS Interconnect," Digital Equipment Corporation, Hudson, MA, December 1986.

F. T. Leighton, "Networks, Parallel Computation and VLSI Design," AMS/MAA National Meeting, January 1987.

F. T. Leighton, "Simulating Special-Purpose Networks with General-Purpose Networks," MIT LIDS Workshop on Distributed Computing, Cambridge, MA, January 1987.

F. T. Leighton, "Some Computational Properties of the Hypercube," BBN, Cambridge, MA, March 1987.

F. T. Leighton, "Some Computational Properties of the Hypercube," Princeton Workshop on Algorithms, Princeton, NJ, March 1987.

J. L. Wyatt, Jr., "Tellegen's Theorem -- What It Says, Why It's True, and Some of the Things It Predicts," Department of Computer Science, California Institute of Technology, Pasadena, CA, March 1987.

VLSI Memo No. 86-348
November 1986

FREQUENCY LIMITATIONS IN CIRCUITS COMPOSED OF LINEAR DEVICES

Lance A. Glasser

Abstract

This paper investigates limitations on the frequency response of networks constructed out of components specified by their small signal models. Tellegen's theorem is used to find the maximum frequency of oscillation. A test on the complex non-Hermitian matrix Y, based on the convexity of the numerical range, is developed to determine if the quadratic form $0 = v^H Y v$ has a nonunique solution for v. We show that $v = 0$ is unique if $Ye^{j\theta} + Y^H e^{-j\theta}$ is positive definite for some θ in the interval $[0,2\pi)$. Transistor and negative resistance amplifier examples are developed.

Microsystems Research Center
Room 39-321
Massachusetts Institute of Technology
Cambridge Massachusetts 02139
Telephone (617) 253-8138

Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency under contract number N00014-80-C-0622.

Author Information

Glasser: Department of Electrical Engineering and Computer Science and Research Laboratory of Electronics, MIT, Room 36-880, Cambridge, MA 02139; (617) 253-4677.

Frequency Limitations in Circuits Composed of Linear Devices[1]

Lance A. Glasser
Department of Electrical Engineering and Computer Science
and the
Research Laboratory of Electronics
Massachusetts Institute of Technology, Room 36–880
Cambridge, Massachusetts 02139
(617) 253–4677

Abstract: This paper investigates limitations on the frequency response of networks constructed out of components specified by their small signal models. Tellegen's theorem is used to find the maximum frequency of oscillation. A test on the complex non-Hermitian matrix $\mathbf{Y}$, based on the convexity of the numerical range, is developed to determine if the quadratic form $0 = \mathbf{v}^H\mathbf{Y}\mathbf{v}$ has a nonunique solution for $\mathbf{v}$. We show that $\mathbf{v} = 0$ is unique iff $\mathbf{Y}e^{j\theta} + \mathbf{Y}^H e^{-j\theta}$ is positive definite for some $\theta \in [0, 2\pi)$. Transistor and negative resistance amplifier examples are developed.

1 Introduction

The frequency limitations of a circuit are determined by the frequency characteristics of its components and the cleverness with which those components are interconnected. It is possible to discover limitations and bounds on the frequency response of a circuit based only on the characteristics of the components. For instance, the natural frequencies (poles and zeros of the admittance) of any one-port constructed only of linear passive resistors and capacitors must lie on the negative real axis of the s-plane ($\omega = 0$ and $\sigma \leq 0$, where $s \triangleq \sigma + j\omega$) [1]. In an example of more relevance today, we can examine a circuit composed of incrementally active devices, such as transistors, and ask, what is the range of natural frequencies achievable by linear time-invariant autonomous circuits built of these components?

Figure 1a illustrates a simple model for a MOS transistor together, in Fig. 1b, with the permitted natural frequencies of oscillation of networks constructed solely of these devices.[2] None of these networks, regardless how cleverly constructed, can have purely sinusoidal natural frequencies of oscillation above $\omega_{\max}$. This example was adapted from the work of Thornton [2, 3]. In Fig. 1b, $R = 0$ and the permitted and forbidden regions are separated by the line

$$\sigma = \frac{g_m^2}{4g_D C_{GS}} - \frac{\omega^2 g_D C_{GS}}{g_m^2}. \tag{1}$$

[1] This research was supported by the Defense Advanced Research Projects Agency under contract number N00014-80-C-0622.

[2] For left-half-plane frequencies the natural frequencies of "oscillation" have voltage solutions that decay exponentially with time.

The reason poles can lie on the $j\omega$ axis without inductors is that the transistors can be used to make gyrators which enable capacitors to emulate inductors. Note how the permitted natural frequencies of oscillation in Fig. 1b reduce to that of an RC network as g_m goes to zero.

The natural frequencies of oscillation of a transistor are closely related to the frequencies at which that transistor can be active. Mason was the first to develop a figure of merit for a transistor in a lossless reciprocal embedding, with his work on the *unilateral gain* of a linear two port [4]. Later, Thornton published his work on the allowed natural frequencies of active RC networks [5], relating it back to Mason's. Kuh and Desoer generalized both works by expanding the class of networks considered and allowing nonreciprocal embeddings [6,7]. In this paper, the class of networks considered will be further expanded, systematized, and extended in a way suggestive of a simple computation to test a complex frequency to determine whether or not it can be a natural frequency of a network composed of any interconnection of components specified by their small signal models. A general statement of the problem is:

> **Given a set of linear components, taken in any multiplicity and impedance scaled by any positive real number, what is the lowest sinusoidal frequency, or more generally the largest region in the s-plane, for which one can prove that connected networks built of these components cannot sustain oscillation.**

The first step is to reformulate the problem in terms of Tellegen's theorem [8] to achieve increased generality. This makes it possible to express the problem in terms of any set of components S, characterized by their small signal models and interconnected by wire. This generalization is important because the performance of most modern devices is dominated by a complex network of parasitics. To neglect these parasitics is to be overly optimistic, such as in the case of Fig. 1 where the use of inductors can increase $\omega_{\max}$ to ∞. The reason $\omega_{\max}$ can be increased without bound in this simple model is that all of the internal capacitors are directly connected across external terminals and hence can be resonated out, at any desired frequency, with a parallel inductor. This clearly nonphysical prediction occurs because the model used was excessively simple. Note that $\sigma_{\max}$ does not increase because both inductors and capacitors absorb power for all time when excited with a growing exponential voltage or current waveform (in this problem, $\omega = 0$ at $\sigma_{\max}$).

The Tellegen formulation leads to a complex nonHermitian matrix $\mathbf{Y}(s)$ which captures the relevant frequency domain information. In this matrix, each type of component need be represented only once. A simple test on $\mathbf{Y}(s)$ will then be presented which can discover whether or not nonzero voltage and current solutions are permitted at a frequency s. From this we can discover regions of the s-plane where, independent of the circuit topology, the network cannot oscillate. Since the synthesis problem is not addressed, one can not use this theory to predict that all regions of the s-plane in which oscillation is permitted are achievable with a realizable circuit. Thus these results have a fundamental limits flavor in which one can say definitively what is forbidden but not what is necessarily realizable.

We have motivated this investigation with the goal of mapping out the s-plane into forbidden and permitted regions. In the next three sections we will focus the discussion on the subproblem of examining a specific point in the plane, s_0, and discovering whether

or not it is in the forbidden region. In Section 5 (Examples) we again look at the s-plane as a whole but, instead of mapping it completely, we will parameterize it in terms of the two points where the line separating the forbidden and permitted regions intersect the real and imaginary axis: $\sigma_{\max}$ and $\omega_{\max}$.

2 Tellegen Formulation and the Conservation of Complex Power

One of the many special cases of Tellegen's theorem states that the inner product of branch voltages $\mathbf{v}$ and branch currents $\mathbf{i}$ is zero. $\mathbf{v}$ is the column vector of Laplace transforms of branch voltages and $\mathbf{i}$ is the column vector of Laplace transforms of branch currents, with associated reference directions imposed. The branches are numbered. We have

$$0 = \mathbf{v}^H(s)\mathbf{i}(s), \tag{2}$$

where $\mathbf{x}^H$ is defined as the complex conjugate transpose of $\mathbf{x}$. Equation (2) can be interpreted as the conservation of complex power. The fact that two real quantities are conserved in (2)—the physically intuitive conservation of real power and the more enigmatic conservation of imaginary power—makes this problem somewhat nonstandard.

Tellegen's theorem for a network can also be expressed in terms of the port or terminal voltages and currents of its subnetworks [9]. Let $S \triangleq \{\mathcal{N}_1, \ldots, \mathcal{N}_M\}$ be any set of linear multiports, not necessarily all of a size, characterized by an associated set of admittance matrices $\mathcal{Y} \triangleq \{\mathbf{Y}_1(s), \ldots, \mathbf{Y}_M(s)\}$. For each component k we have

$$\mathbf{i}_k = \mathbf{Y}_k \mathbf{v}_k. \tag{3}$$

Let $\mathcal{M}$ be any network obtained by producing a connected network from the elements of S and ideal wire, without violating the port assumptions of the admittance matrices. Tellegen's theorem for $\mathcal{M}$ states

$$0 = \sum_{k=1}^{M} \mathbf{v}_k^H \mathbf{i}_k \tag{4}$$

or

$$0 = \sum_{k}^{M} \mathbf{v}_k^H \mathbf{Y}_k \mathbf{v}_k. \tag{5}$$

It is helpful to rewrite (5) in block diagonal form

$$0 = \mathbf{x}^H \mathbf{Y} \mathbf{x}, \tag{6}$$

where

$$\mathbf{x}^T \triangleq (\mathbf{v}_1^T, \ldots, \mathbf{v}_M^T) \tag{7}$$

and

$$\mathbf{Y}(s) \triangleq \begin{pmatrix} \mathbf{Y}_1 & 0 & 0 \\ 0 & \mathbf{Y}_k & 0 \\ 0 & 0 & \mathbf{Y}_M \end{pmatrix}. \tag{8}$$

Equation (6) is a quadratic form but note that $\mathbf{Y}$ is typically nonHermitian. A physical interpretation of $\mathbf{Y}$ is shown in Fig. 2. The network it represents can be thought of as a compound component containing one of each different subcomponent. By wiring its ports one obtains a network $\mathcal{M}$.

In deference to the leverage of integrated circuit technology, it is greatly desirable to generalize the set of networks considered by the theory to networks which include, not just one, but any number of instantiations of the components from the set $\mathcal{S}$. Let $\mathcal{G} \triangleq \{\mathcal{L}_1, \ldots, \mathcal{L}_N\}$ be any set of multiports characterized by admittance matrices that are non-negative scalar multiples of matrices from the set $\mathcal{Y}$. Let $\mathcal{N}$ be any connected network obtained by interconnecting $\mathcal{L}_1, \ldots, \mathcal{L}_N$ using using only ideal wire and ideal transformers. Let $a_1, ..., a_N \in R^N$ represent the non-negative admittance scaling factors on the admittance matrix from $\mathbf{Y}$. Tellegen's theorem for $\mathcal{N}$ can be written

$$0 = \sum_i^N a_i \mathbf{x}_i^H(s)\mathbf{Y}(s)\mathbf{x}_i(s). \tag{9}$$

Equation (9) contains an explicit summation over compound component instantiations and an implicit sum of subcomponent types. The physical interpretation of (9) is one of multiple, admittance scaled, instantiations of the network in Fig. 2, wired together in an arbitrary manner. Subcomponents which are not needed have their ports shorted. With a change of variables, (9) may be rewritten

$$0 = \sum_i^N \mathbf{w}_i^H(s)\mathbf{Y}(s)\mathbf{w}_i(s), \tag{10}$$

where

$$\mathbf{w}_i(s) \triangleq \sqrt{a_i}\mathbf{x}_i(s). \tag{11}$$

3 Main Result

Limitations on the frequency behavior of $\mathcal{N}$ can be discovered by investigating the frequencies s at which (9) has nontrivial voltage solutions. No network $\mathcal{N}$ can be constructed to have a natural frequency $s_0 \in C$ unless there is a nonzero voltage solution to (9). Using this criterion we can divide the s-plane into forbidden and permitted regions; see Fig. 1b.

It is desirable to have a simple test on the set of admittance matrices $\mathcal{Y}$ which tells whether s_0 is in the permitted or forbidden region. If $\mathbf{Y}$ was Hermitian (or even antiHermitian) then testing to see if (6), a special case of (9), has a nonzero solution $\mathbf{v}$ would be straightforward. If $\mathbf{Y}$ is either positive- or negative-definite then no nontrivial solutions exist. Standard tests for definiteness include looking at the eigenvalues, pivots, or subdeterminates of $\mathbf{Y}$ [10].

We can examine necessary conditions for (6) to have a solution by looking separately at the Hermitian and antiHermitian parts of (6). Let

$$\mathbf{Y}_H \triangleq \frac{1}{2}(\mathbf{Y} + \mathbf{Y}^H) \tag{12}$$

and

$$\mathbf{Y}_{AH} \triangleq \frac{1}{2}\left(\mathbf{Y} - \mathbf{Y}^H\right), \tag{13}$$

where $\mathbf{Y}_H$ is the Hermitian part of $\mathbf{Y}$ and $\mathbf{Y}_{AH}$ is the antiHermitian part. Testing these parts separately, however, generates an overly optimistic prediction of the size of the permitted region because, for instance, when looking at $\mathbf{Y}_H$ we are allowing $\mathbf{v}^H\mathbf{Y}_{AH}\mathbf{v}$ to take on any (imaginary) value. Said another way, if $\mathcal{V}_h$ is the solution set to $0 = \mathbf{v}_h^H\mathbf{Y}_H\mathbf{v}_h$, excluding $\mathbf{v}_h = 0$, and $\mathcal{V}_{ah}$ the solution set to $0 = \mathbf{v}_{ah}^H\mathbf{Y}_{AH}\mathbf{v}_{ah}$, excluding $\mathbf{v}_{ah} = 0$, the solution set to (6) is the intersection of $\mathcal{V}_h$ and $\mathcal{V}_{ah}$, which can easily be empty even when $\mathcal{V}_h$ and $\mathcal{V}_{ah}$ are nonempty. In (6) we are looking for simultaneous solutions to two quadratic forms.

While definite tests on the Hermitian part of $\mathbf{Y}$ is not quite what we want to do, it is close. The following theorem is the basis for a simple method of computing the boundary separating the forbidden and permitted regions.

Theorem 1 (Main result)

Let $\mathcal{Y} \triangleq \{\mathbf{Y}_1(s), ..., \mathbf{Y}_M(s)\}$ be any finite set of admittance matrices (not necessarily of the same size). Let $\mathcal{L}_1, ..., \mathcal{L}_N$ be any set of linear multiports characterized by admittance matrices that are non-negative scalar multiples of matrices from the set $\mathcal{Y}$; i.e., for each $i \in \{1, ..., N\}$ the admittance matrix of $\mathcal{L}_i$ is of the form $a_i\mathbf{Y}_k(s)$ for some $k \in \{1, ..., M\}$, where $a_i \in R$ is non-negative. Define $\mathbf{Y}(s) \triangleq \text{diag}\{\mathbf{Y}_1(s), ..., \mathbf{Y}_M(s)\}$. Let $\mathcal{N}$ be any connected network obtained by interconnecting $\mathcal{L}_1, ..., \mathcal{L}_N$ using only ideal (multiwinding) transformers and ideal connecting wire, and $s_0 \in C$ be any complex frequency not a pole or zero of $\mathbf{Y}(s)$. If

$$\mathbf{A}(\theta, s_0) \triangleq \frac{1}{2}\left(\mathbf{Y}(s_0)e^{j\theta} + \mathbf{Y}^H(s_0)e^{-j\theta}\right) \tag{14}$$

is positive definite for some $\theta \in [0, 2\pi)$, then s_0 is not a natural frequency of $\mathcal{N}$. If $A(\theta, s_0)$ is not positive definite for any $\theta \in [0, 2\pi)$ then there exists a nonzero voltage solution $\hat{\mathbf{x}}_1(s_0), ..., \hat{\mathbf{x}}_N(s_0)$ to the Tellegen statement of conservation of complex power

$$0 = \sum_i^N a_i\mathbf{x}_i^H(s_0)\mathbf{Y}(s_0)\mathbf{x}_i(s_0). \tag{9}$$

Note that the relatively straightforward test described above suffices to demonstrate that s_0 is not a natural frequency of a remarkably large class of networks: the class of all networks made of interconnections of any number of multiports described by admittance matrices in the set $\mathcal{Y}$, or positive scalar multiples of admittance matrices in $\mathcal{Y}$. There is no requirement that $N \leq M$, i.e., the number of elements in the network is unlimited.

A restatement of Theorem 1 is more suggestive of an algorithm: If $\mathbf{Y}_k(s_0)e^{j\theta} + \mathbf{Y}_k^H(s_0)e^{-j\theta}$, the Hermitian part of each phase-shifted subcomponent admittance matrix, is positive definite for all $\mathbf{Y}_k(s_0) \in \mathcal{Y}$ at the same θ then no network made from the elements of $\mathcal{Y}$, taken in any multiplicity and admittance scaled by any non-negative real number, can have a natural frequency $s_0 \in C$.

4 Proof of Main Result

The proof of Theorem 1 rests on Proposition 1 below, which is a special case of Theorem 1.

Proposition 1

Let $\mathcal{L}_1, ..., \mathcal{L}_M$ be linear multiports characterized by matrices $\mathbf{Y}_1(s), ..., \mathbf{Y}_M(s)$, respectively. Let $\mathcal{M}$ be any network created by interconnecting $\mathcal{L}_1, ..., \mathcal{L}_M$ using only ideal (multiwinding) transformers and ideal connecting wire. Define

$$\mathbf{Y}(s) \triangleq \text{diag}\{\mathbf{Y}_1(s), ..., \mathbf{Y}_M(s)\} \tag{8}$$

and

$$\mathbf{A}(\theta, s_0) \triangleq \frac{1}{2}\left(\mathbf{Y}(s_0)e^{j\theta} + \mathbf{Y}^H(s_0)e^{-j\theta}\right) \tag{14}$$

as before. For a complex frequency s_0, not a pole or zero of $\mathbf{Y}(s)$, s_0 is not a natural frequency of $\mathcal{M}$ if $A(\theta, s_0)$ is positive definite for some $\theta \in [0, 2\pi)$.

Note that the assumptions of Proposition 1 are more restrictive than those of Theorem 1 in that the network elements of $\mathcal{M}$ are precisely those with admittance matrices $\mathbf{Y}_1(s), ..., \mathbf{Y}_M(s)$. $\mathcal{M}$ contains exactly one element with admittance matrix $\mathbf{Y}_1(s)$, one element with admittance matrix $\mathbf{Y}_2(s)$, etc. The proof of Proposition 1 rests on Lemma 1 below.

Lemma 1

Let $\mathbf{B} \in C^{n \times n}$ be any complex matrix. The solution $\mathbf{v} = 0$ of

$$0 = \mathbf{v}^H \mathbf{B} \mathbf{v} \tag{15}$$

is unique iff

$$\mathbf{A}(\theta) \triangleq \frac{1}{2}\left(\mathbf{B}e^{j\theta} + \mathbf{B}^H e^{-j\theta}\right) \tag{16}$$

is positive definite for some $\theta \in [0, 2\pi)$.

The proof of Lemma 1 rests on the definitions and facts below.

Definition 1

The *numerical range* of a complex matrix $\mathbf{B} \in C^{n \times n}$ is the set $W(\mathbf{B})$ of all complex numbers of the form $\mathbf{x}^H \mathbf{B} \mathbf{x}$, where $\mathbf{x}$ varies over all vectors on the unit sphere, $\mathbf{x}^H \mathbf{x} = 1$ [11–14].

Figure 3a illustrates a possible numerical range for $\mathbf{Y}$. This case is an example of a situation in which the Hermitian and antiHermitian parts of $\mathbf{Y}$ test as indefinite yet $0 \notin W(\mathbf{Y})$. In the figure, λ_{minH} and λ_{maxH} represent the smallest and largest eigenvalues of $\mathbf{Y}_H$ and λ_{minAH} and λ_{maxAH} represent the smallest and largest eigenvalues of $\mathbf{Y}_{AH}$, respectively. Fig. 3b illustrates a rotation of $\mathbf{Y}$ which causes $\mathbf{A}(\theta)$ to test positive definite.

Definition 2

A subset $\mathcal{G} \in C$ is said to be *convex* if for $\lambda \in R$, $(1 - \lambda)x + \lambda y \in \mathcal{G}$ whenever $x \in \mathcal{G}$, $y \in \mathcal{G}$, and $0 < \lambda < 1$.

Fact 1

The Toeplitz-Hausdorff theorem states the remarkable fact that the numerical range of a matrix is always convex [15].

Lemma 1 rests on Lemmas 2 and 3 below.

Lemma 2

Let $\mathbf{B} \in C^{n \times n}$ be any complex matrix. The solution $\mathbf{v} = 0$ of $0 = \mathbf{v}^H \mathbf{B} \mathbf{v}$ is unique iff $0 \notin W(\mathbf{B})$.

Proof of Lemma 2

Let $\|\mathbf{v}\|^2$ denote the L_2 inner product $\mathbf{v}^H \mathbf{v}$ and let S denote the unit sphere in C^n, i.e., $S \triangleq \{\mathbf{v} \in C^n \mid \|\mathbf{v}\| = 1\}$. Clearly (15) has a nonzero solution $\hat{\mathbf{v}}$ iff (15) has a solution $\mathbf{v} \triangleq \hat{\mathbf{v}}/\|\hat{\mathbf{v}}\|$ lying in S. Since $\mathbf{v} = 0$ is always a solution to (15), it is unique iff $0 \notin W(\mathbf{B})$. ∎

Lemma 3

A closed convex set K in the complex plane does not contain the origin iff there exists a rotation about the origin that carries all of K into the the open right-half-plane.

Discussion of Lemma 3

This Lemma is obvious but the interested reader could construct a proof by noting that two closed convex sets, in our case K and the origin, containing no points in common must lie on opposite sides of a separating hyperplane (in this case a straight line) [16]. To show that a rotation about the origin carries all of K into the open right-half-plane it is sufficient to show that there exists a rotation which brings the line into the open right half plane, parallel to the imaginary axis, with the origin (which hasn't moved) on one side and K on the other.

Proof of Lemma 1

By Lemma 2, (15) has a nonzero solution $\mathbf{v} \in C^n$ iff $0 \in W(\mathbf{B})$. Equivalently, $0 \notin W(\mathbf{B})$ iff $W(\mathbf{B}e^{j\theta})$ is a subset of the open right-half plane for some $\theta \in [0, 2\pi)$, by Lemma 3. Let $\mathbf{D} \triangleq \mathbf{B}e^{j\theta}$ and $\mathbf{A} \triangleq (\mathbf{D} + \mathbf{D}^H)/2$ be the Hermitian part of $\mathbf{D}$. For any matrix $\mathbf{D} \in C^{n \times n}$, $W(\mathbf{A})$ is the projection onto the real axis of $W(\mathbf{D})$. The trivial solution to (15) is unique iff $W(\mathbf{A}, \theta)$ is contained in the strictly positive reals, i.e., iff $\mathbf{A}(\theta)$ is positive definite for some $\theta \in [0, 2\pi)$. ∎

Proof of Proposition 1

Proposition follows directly from Lemma 1 and Tellegen's Theorem, where we have identified the block diagonal matrix $\mathbf{Y}(s)$ of (6) with the matrix $\mathbf{B}$ of Lemma 1. ∎

The generalization of Proposition 1 to Theorem 1 requires the fact below.

Fact 2

Define $\mathbf{B} \in C^{n \times n}$ as the block diagonal matrix $\mathbf{B} \triangleq \text{diag}(\mathbf{B}_1, ..., \mathbf{B}_N)$ where $\{\mathbf{B}_1, ..., \mathbf{B}_N\}$ is a set of complex square matrices. $W(\mathbf{B})$ is the convex hull of the set of numerical ranges $\{W(\mathbf{B}_1), ..., W(\mathbf{B}_N)\}$ [15].

Note that $W(\mathbf{B})$ represents the numerical range of the sum of quadratic forms $\sum \mathbf{x}_i^H \mathbf{B}_i \mathbf{x}_i$ where $\sum \|\mathbf{x}_i\|^2 = 1$. Clearly, for any complex matrix, $W(\text{diag}(\mathbf{B}, ..., \mathbf{B})) = W(\mathbf{B})$.

Proof of Theorem 1

From Tellegen's Theorem, no network $\mathcal{N}$ can have a natural frequency s_0 if (9) has a unique solution $\mathbf{x}_1, ..., \mathbf{x}_N = 0$. With a change of variable, (9) has a unique solution at the origin iff (10) has a unique solution at the origin (unless all of the admittance scaling factors a_i are zero, in which case there is no network). From Lemma 1, (10) has a nonzero solution iff $0 \in W(\text{diag}(\mathbf{Y}, ..., \mathbf{Y}))$, but from Fact 2, $W(\text{diag}(\mathbf{Y}, ..., \mathbf{Y})) = W(\mathbf{Y})$. The remainder the the proof follows in the same manner as the proof of Proposition 1. ∎

Further insight into this theorem may be gained by looking at the Hermitian parts of the rotated admittance matrices in $\mathcal{Y}$. Let $\mathcal{A} \triangleq \{\mathbf{A}_1, ..., \mathbf{A}_M\}$ be a finite set of Hermitian matrices in one to one correspondence with the admittance matrices in $\mathcal{Y}$. Let

$$\mathbf{A}_k(\theta, s_0) \triangleq \frac{1}{2}(\mathbf{Y}_k e^{j\theta} + \mathbf{Y}_k^H e^{-j\theta}) \tag{17}$$

for all $\mathbf{Y}_k \in \mathcal{Y}$. Let $\hat{\theta} \in [0, 2\pi)$ be a rotation for which $\mathbf{A}(\hat{\theta}, s_0)$ is positive definite. By Lemmas 1 and 2, $W(\mathbf{Y}e^{j\hat{\theta}})$ lies entirely in the right half plane and by Fact 2 each $W(\mathbf{Y}_i e^{j\hat{\theta}})$ also lies in the right half plane for all $\mathbf{Y}_i \in \mathcal{Y}$. Therefore each $\mathbf{A}_k(\hat{\theta}, s_0)$ is positive definite for all $\mathbf{A}_k \in \mathcal{A}$. Thus all subcomponents used in $\mathcal{N}$ (those in which $a_i > 0$) result in a strictly positive real contribution to the sum $\sum a_i \mathbf{x}_i^H \mathbf{A} \mathbf{x}_i$ unless their port voltages are zero. The only way for this sum to equal zero, as it must by Tellegen's Theorem, is if all port voltages are zero.

It is worth observing that the first half of the Theorem—the part which details a sufficient test to prove that s_0 is not a naturally frequency of $\mathcal{N}$—can trivially be derived from (6) by multiplying each side of the equality by $\exp(j\theta)$ and taking the real part. It is the second half of the theorem—the part that states that if a θ cannot be found to make $\mathbf{A}(\theta, s_0)$ positive definite then a nonzero voltage solution for (9) exists—that requires the convexity of the numerical range. In other words, this paper is concerned with investigating limits no less strict than those which arise from the conservation of real and imaginary power.

When solutions to (6) are found at the same s that $\mathbf{Y}$ is singular, further inspection of the result is required. When $\mathbf{Y}$ is singular, voltage solutions $\mathbf{v}$ can be sustained with zero current. These solutions, in which no real or reactive power flow through the network, are generally of little interest from the standpoint of this work. For the same reason, when formulating $\mathbf{Y}$, the definite form of the admittance matrix must be used.

5 Examples

In this section we will go through three examples to illustrate the scope and utility of the theory. The first example is a negative-resistance reflection amplifier constructed from the three elements illustrated in Fig. 4. Figure 4a shows a negative resistance device with capacitive and resistive parasitics. This problem is best formulated in terms of impedances rather than admittances. The impedance of the amplifier is

$$Z_{\text{amp}}(s) = R_S + \frac{1}{sC - G}. \tag{18}$$

Fig. 4b illustrates the inductors available to tune out the reactance of Z_{amp}. The inductors have a parasitic series resistance R_L, where $\omega_L \triangleq R_L/L$ is a constant of the technology. We have

$$Z_L(s) = R_L + sL = (\omega_L + s)L. \tag{19}$$

The third element type we have in the circuit, also illustrated in Fig. 4b, is the passive resistor: $Z_R(s) = R_R$. The $\mathbf{Z}$ matrix for $\mathcal{S}$ is

$$\mathbf{Z} = \begin{pmatrix} Z_{\text{amp}} & 0 & 0 \\ 0 & Z_L & 0 \\ 0 & 0 & Z_R \end{pmatrix}. \tag{20}$$

Since $\mathbf{Z}$ is diagonal, the eigenvalues appear on the diagonal and the eigenvectors are orthogonal. Solving for $W(\mathbf{Z})$,

$$W(\mathbf{Z}) = Z_{\text{amp}}\|\mathbf{x}_1\|^2 + Z_L\|\mathbf{x}_2\|^2 + Z_R\|\mathbf{x}_3\|^2, \tag{21}$$

where

$$\|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 + \|\mathbf{x}_2\|^2 = \|\mathbf{x}\|^2 = 1. \tag{22}$$

$\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are the eigenvectors of $\mathbf{Z}$. Figure 5a illustrates the numerical range of $\mathbf{Z}$. The dashed lines indicate how Z_{amp} and Z_L move with increasing ω ($\sigma = 0$). After some algebra, one finds

$$\omega_{\max} = \frac{1}{C}\sqrt{\frac{G}{R_S} - G^2 - \frac{\omega_L C}{R_S}}. \tag{23}$$

This solution is found at $a_{11}(\theta) = a_{22}(\theta) = 0$. In general, $\det \mathbf{A}(\theta)$ can be represented as a Fourier series with frequency components from $-N$ to N where N is the rank of $\mathbf{A}$. Computational advantage can be gained by also noting that

$$\det\mathbf{A} = \prod_{k=1}^{M} \det\mathbf{A}_k. \tag{24}$$

With $R_L = 0$, (26) reduces to the more optimistic expression for $\omega_{\max}$ one would obtain by simply examining the Hermitian part of $\mathbf{Z}$. With the element values of Table 1, $\omega_{\max} = 2$ is predicted if complex power is conserved, but $\omega_{\max} = 3$ is predicted if only real power is conserved (i.e., looking at only the Hermitian part of $\mathbf{Z}$ or only $\theta = 0$).

element	realistic inductor	ideal inductor	units
R_S	0.1	0.1	Ω
G	1	1	Ω^{-1}
C	1	1	F
R_L	0.5	0	Ω
L	1	1	H
R_R	1	1	Ω
ω_L	0.5	0	s^{-1}
$\omega_{\max}$	2	3	s^{-1}

Table 1

The numerical range of $\mathbf{A}(\theta)$ is real; it is illustrated in Fig. 5b for $\omega = 2.5$. It lies between the minimum and maximum eigenvalues;

$$\lambda_{\min}(\mathbf{A}) = \inf \operatorname{Re} \begin{cases} Z_{\text{amp}} e^{j\theta} \\ Z_L e^{j\theta} \\ Z_R e^{j\theta} \end{cases} \tag{25}$$

and

$$\lambda_{\max}(\mathbf{A}) = \sup \operatorname{Re} \begin{cases} Z_{\text{amp}} e^{j\theta} \\ Z_L e^{j\theta} \\ Z_R e^{j\theta} \end{cases} . \tag{26}$$

Note that $\lambda_{\min}(\mathbf{A}(\theta))$ can contain local maxima and points at which the derivative with respect to θ is undefined.

The second example we consider is interconnections of the three-terminal MOS transistor illustrated in Fig. 1a. For this circuit,

$$\mathbf{Y} = \begin{pmatrix} \frac{sC}{sRC+1} & 0 \\ \frac{g_m}{sC+1} & g_D \end{pmatrix} . \tag{27}$$

Note that the use of the source node as the datum does not imply any restriction to common source configurations. A datum must be chosen so that a definite admittance matrix can be developed. An indefinite admittance matrix would always admit nonzero v solutions since an arbitrary constant could be added to all node voltages. Note that the numerical range of any 2×2 matrix is a closed elliptical disk with foci at the eigenvalues.

To aid our pursuit of closed-form solutions, we separate $\mathbf{Y}$ into Hermitian and anti-Hermitian parts. Rewriting $\mathbf{A}(\theta)$,

$$\mathbf{A}(\theta) = \cos\theta \left(\mathbf{Y}_H + j\mathbf{Y}_{AH} \tan\theta\right) . \tag{28}$$

For our purposes, the $\cos\theta$ scaling factor on $\mathbf{A}(\theta)$ is inconsequential and may be neglected. Solutions for $\mathbf{A}$ at $\cos\theta = 0$ are generally not of interest because this would imply that the conservation of real power was unimportant. We define

$$\mathbf{A}'(\delta) = \frac{1}{2}(\mathbf{Y}_H + j\delta\mathbf{Y}_{AH}), \tag{29}$$

where $\delta \triangleq \tan\theta$. For $s = \sigma$ we obtain

$$\mathbf{A}'(\delta) = \frac{1}{2}\begin{pmatrix} \frac{2\sigma C}{\sigma RC+1} & (1+j\delta)\frac{g_m}{\sigma RC+1} \\ (1-j\delta)\frac{g_m}{\sigma RC+1} & 2g_D \end{pmatrix} \tag{30}$$

and for $s = j\omega$ we obtain

$$\mathbf{A}'(\delta) = \frac{1}{2}\begin{pmatrix} \frac{2\omega^2C^2R+2\omega C\delta}{1+(\omega RC)^2} & (1+j\delta)\frac{g_m}{1-j\omega RC} \\ (1-j\delta)\frac{g_m}{1+j\omega RC} & 2g_D \end{pmatrix}. \tag{31}$$

Since $a'_{22} > 0$, the determinant test for positive definiteness only requires examining the $\det\mathbf{A}'$. Note that this was not the case in the reflection amplifier example. $\det\mathbf{A}'$ is quadratic in δ. Solving for $\det\mathbf{A}' = 0$ (the frequency at which $\mathbf{A}'$ becomes positive definite) leads to

$$\sigma_{\max} = \frac{-1 \pm \sqrt{1 + \frac{(1+\delta^2)g_m^2R}{g_D}}}{2RC} \tag{32}$$

and

$$\omega_{\max} = \frac{-\delta \pm \sqrt{\delta^2 + \frac{g_m^2R}{g_D}(1+\delta^2)}}{2RC}. \tag{33}$$

Optimizing with respect to δ to find the min-max values for $\sigma_{\max}$ and $\omega_{\max}$,

$$\sigma_{\max} = \frac{-1 + \sqrt{1 + \frac{g_m^2R}{g_D}}}{2RC} \tag{34}$$

and

$$\omega_{\max} = \frac{g_m^2}{2g_DC}\left(1 + \frac{g_m^2R}{g_D}\right)^{-1/2}. \tag{35}$$

Unlike in the reflection amplifier case, decomposing $\mathbf{Yv}$ into its eigenvalues and eigenvectors is unproductive because the eigenvectors are nonorthogonal. The only useable structure we have is that $W(\mathbf{Y})$ contains the spectrum of $\mathbf{Y}$.

The third example is an extension of the last. Here, realistic parasitics and a body terminal are added to the model of Fig. 1a. The new model, which still does not take into

account distributed effects or source and drain resistances, is shown in Fig. 6. Since the simple model of Fig. 1a is a special case of the model of Fig. 6, the two can be compared.

Parameter	Fig. 1a	Fig. 6	Trivial	Units
g_m	7×10^{-4}	7×10^{-4}	7×10^{-4}	Ω^{-1}
g_{mb}	0	2×10^{-5}	0	Ω^{-1}
C_{GS}	10^{-14}	10^{-14}	10^{-14}	F
C_{GD}	0	3×10^{-15}	0	F
C_{BS}	0	6×10^{-15}	0	F
C_{BD}	0	6×10^{-15}	0	F
C_{GB}	0	10^{-15}	0	F
g_D	8×10^{-5}	8×10^{-5}	0	Ω^{-1}
R_G	1000	1000	0	Ω
$\omega_{\max}$	1.14×10^{11}	2.95×10^{10}	∞	s^{-1}
$\sigma_{\max}$	8.34×10^{10}	1.91×10^{10}	∞	s^{-1}
$f_{\max} = \omega_{\max}/2\pi$	18.1	4.69	∞	GHz
f_T	11.1	7.95	11.1	GHz

Table 2

Table 2 gives the parameter values for the simplified (Fig. 1a) and full (Fig. 6) models. The parameter values are typical of MOSFETs found in VLSI circuits. The Fig. 6 values of $\sigma_{\max}$ and $\omega_{\max}$ were obtained by computer. Note the large differences in speed predicted by the two models. For comparison, the transition frequencies f_T of the two cases are also given. f_T is the frequency at which the magnitude of the output short-circuit common-source current gain drops to 1. The f_T is

$$2\pi f_T = \frac{g_m}{C_{GS} + C_{GD} + C_{GB}}. \tag{36}$$

Note, however, that the f_T of a transistor, while of proven usefulness, is not fundamental in the sense that $\omega_{\max}$ is fundamental, since it assumes a topology. This is why the series gate resistor R_G does not appear in (36) despite its obvious importance to most practical circuits. The shortcomings of f_T are accentuated in the fourth column of Table 2. While the f_T is finite, the maximum frequency of oscillation is infinite. A circuit which can realize current gain at any frequency is illustrated in Fig. 7. The trick to this circuit is that the gate capacitances are added in series while the drain currents are added in parallel. By using many transistors one can make the effective f_T as high as desired. The fourth column represents an exceptional case. More often, as in column three, the f_T is too optimistic a *predictor of circuit performance.*

Since $\omega_{\max}$ includes the effects of *all* parasitics, I believe that it will find great utility in the comparison and development of competing technologies—e.g., GaAs MOSFETs and Si bipolar transistors.

6 Acknowledgements

It is my pleasure to acknowledge helpful and stimulating conversations with Professors John Wyatt, Paul Penfield, Jr., and Gilbert Strang. The work of Prof. Richard Zippel and Mr. John Wroclawski on incorporating this theory into Schema [17] so that complex examples can be symbolically and numerically computed is gratefully acknowledged.

Figures

References

[1] E.A. Guillemin, *Synthesis of Passive Networks*, J. Wiley, N.Y., Chapter 1, 1957.

[2] R.D. Thornton, "Some Limitations of Linear Amplifiers," Sc.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1957.

[3] R.D. Thornton, D. DeWitt, P.E. Gray, and E.R. Chenette, *Characteristics and Limitations of Transistors*, J. Wiley, NY, 1966, Chapter 3.

[4] S.J. Mason, "Power Gain in Feedback Amplifiers," *IRE Trans. Circuit Theory*, Vol. CT-1, No. 2, pp. 20–25, 1954.

[5] R.D. Thornton, "Active RC Networks," *IRE Trans. Circuit Theory*, Vol. CT-4, No. 3, pp. 78–69, September 1957.

[6] E.S. Kuh, "Regenerative Modes of Active Networks," *IRE Trans. Circuit Theory*, Vol. CT-7, No. 1, pp. 62–63, March 1960.

[7] C.A. Desoer and E.S. Kuh, "Bounds on Natural Frequencies of Linear Active Networks," *Proc. Symposium on Active Networks and Feedback Systems*, pp. 415–436, Polytechnic Institute of Brooklyn, April 19–21, 1960.

[8] P. Penfield, Jr., R. Spence, S. Duinker, *Tellegen's Theorem and Electrical Networks*, MIT Press, Cambridge, MA, 1970.

[9] J.L. Wyatt and G. Papadopolos, "Kirchhoff's Laws and Tellegen's Theorem for Networks and Continuous Media," *IEEE Trans. Circuits and Systems*, vol. CAS-31, no. 7, pp. 657–661, Theorem 2, July 1984.

[10] G. Strang, *Linear Algebra and Its Applications*, Academic Press, N.Y., Chapter 6, 1980.

[11] M.H. Stone, *Linear Transformations in Hilbert Space and Their Applications to Analysis*, American Mathematical Soc., N.Y., pp. 130–134, 1932.

[12] C.R. Putnam, *Commutation Properties of Hilbert Space Operators and Related Topics*, Springer-Verlag, N.Y., pp. 8–9, 1986.

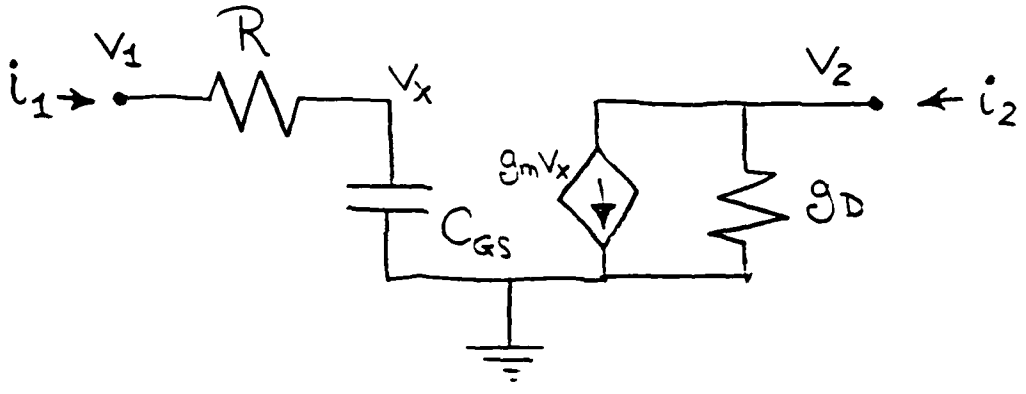
[13] F.F. Bonsall and J. Duncan, *Numerical Ranges of Operators on Normed Spaces and of Elements of Normed Algebras*, Cambridge University Press, Cambridge, G.B., 1971.

[14] F.F. Bonsall and J. Duncan, *Numerical Ranges II*, Cambridge University Press, Cambridge, G.B., 1973.

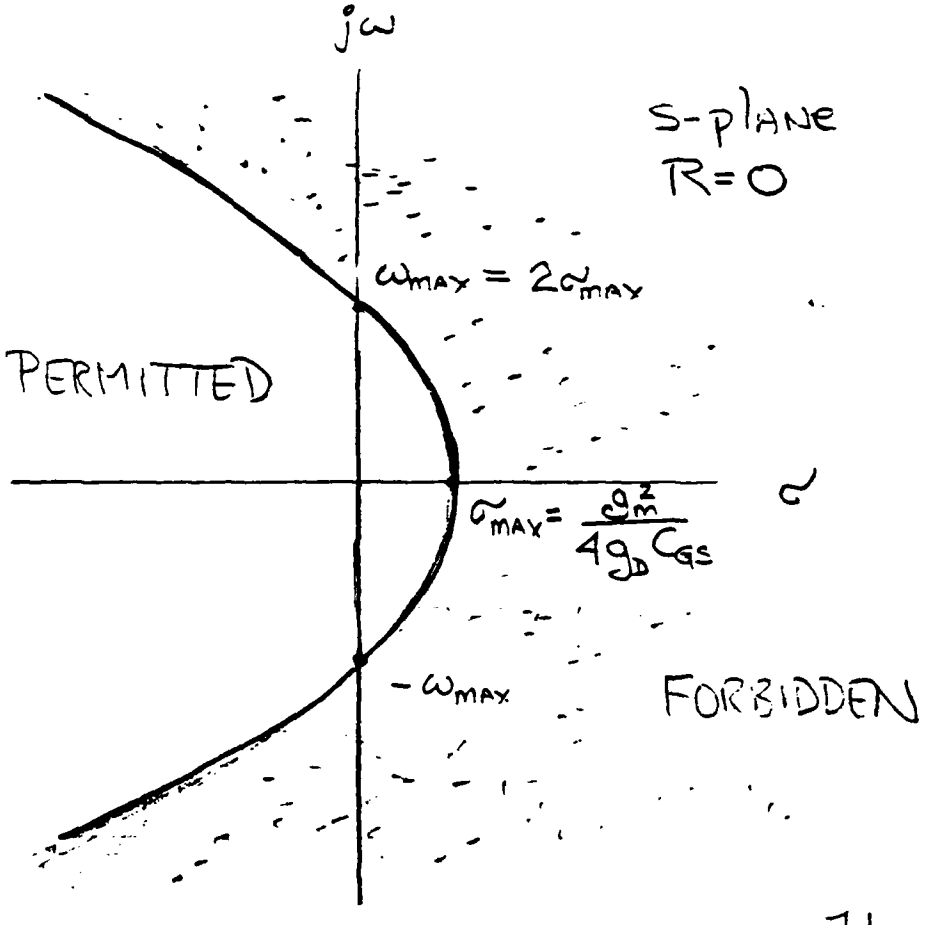
[15] P.R. Halmos, *A Hilbert Space Problem Book*, Chapter 17, Van Nostrand, Princeton, NJ, 1967.

[16] R.T. Rockafellar, *Convex Analysis*, pp. 95–101, Princeton U. Press, Princeton, NJ, 1970.

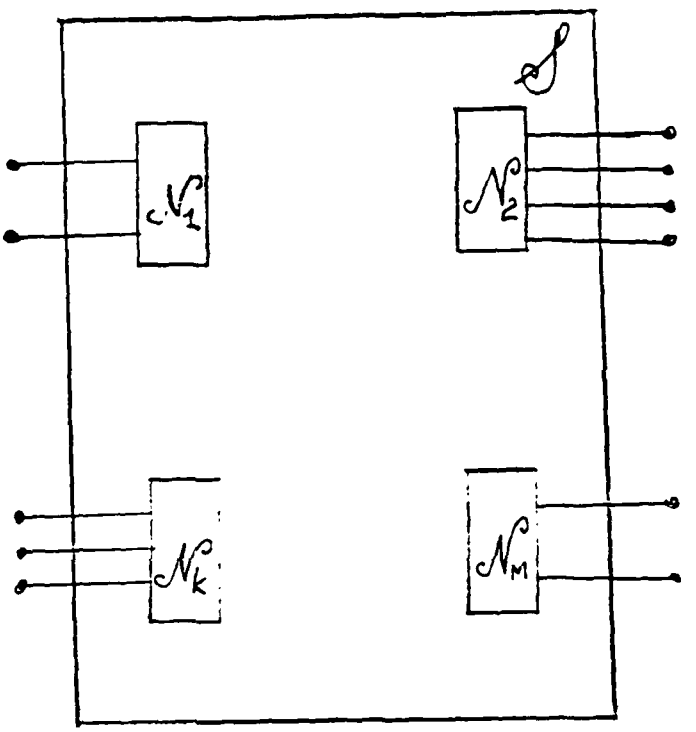
[17] G. Clark and R. Zippel, "Schema: An Architecture for Knowledge Based Design," *International Conference on Computer-Aided Design 85*, pp. 50–52, Santa Clara CA, November 18–21, 1985.

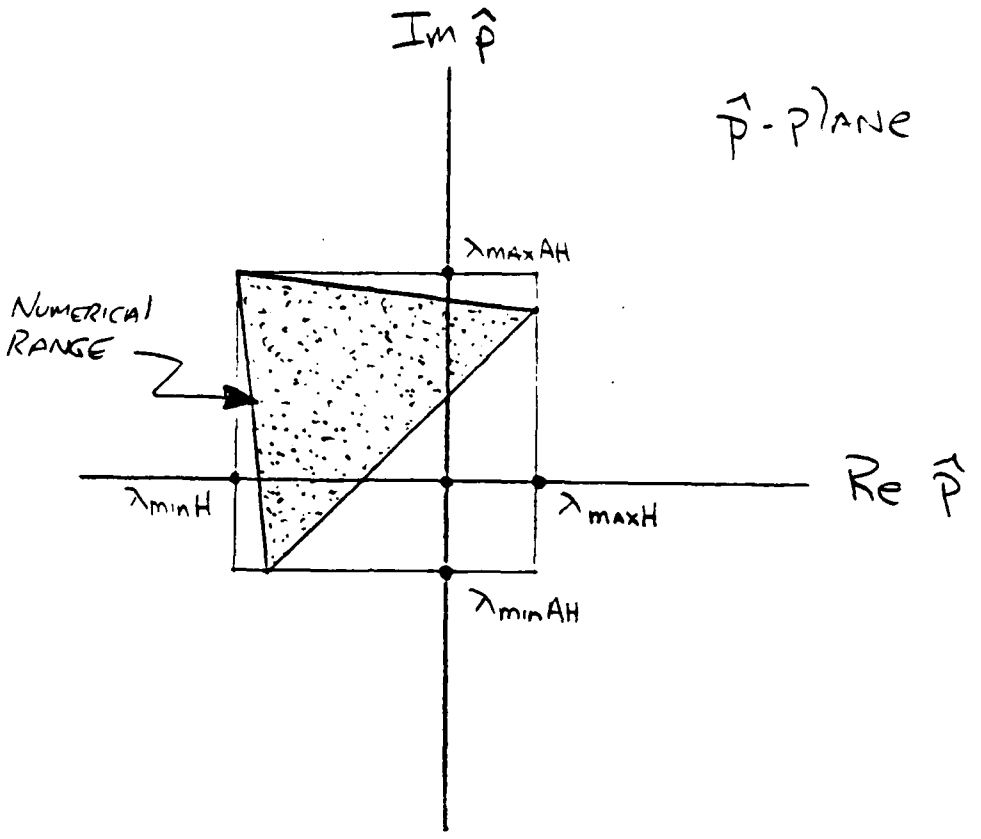

i_1
V_1
R
V_x
V_2
i_2
$g_m V_x$
C_{GS}
g_D

1a

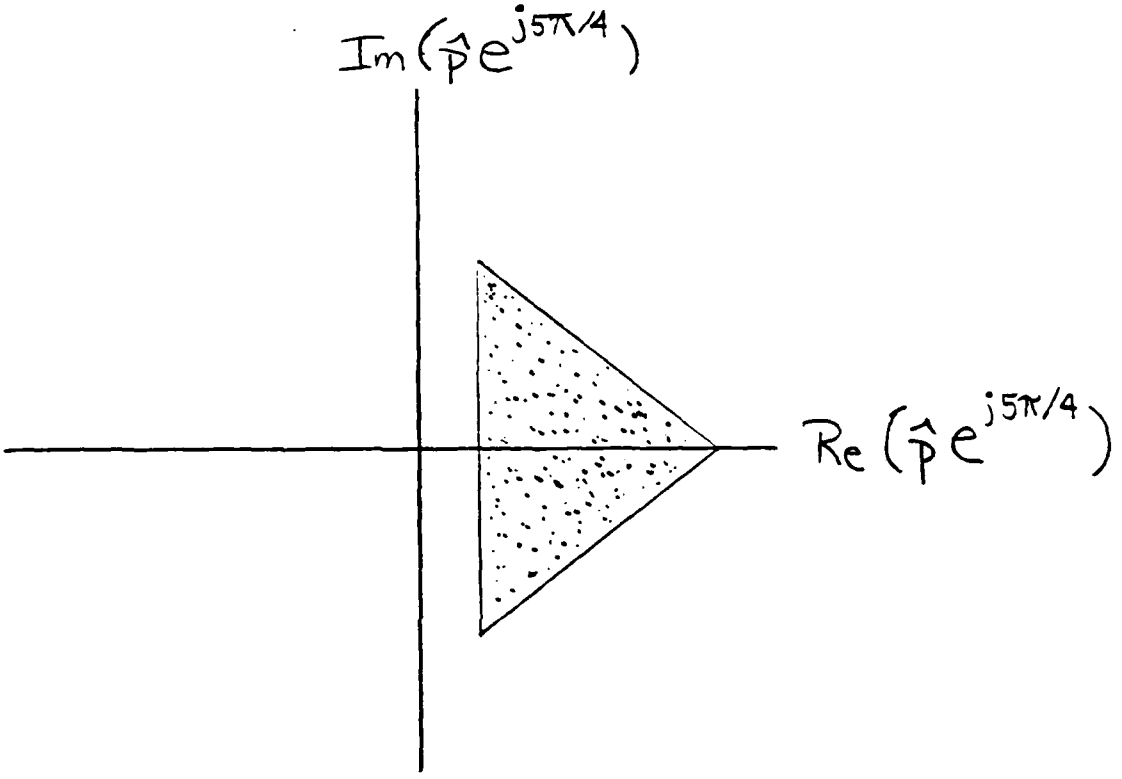

jω
S-PLANE
R=0
$\omega_{MAX} = 2\sigma_{MAX}$
PERMITTED
$\sigma_{MAX} = \frac{g_m^2}{4 g_D C_{GS}}$
σ
$-\omega_{MAX}$
FORBIDDEN

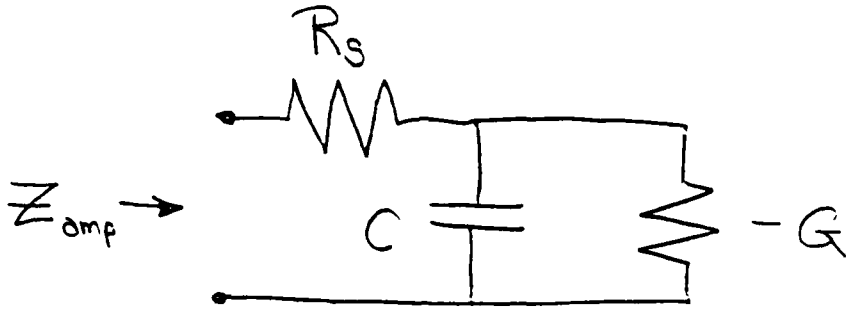
1b


$\mathcal{S}$
$\mathcal{N}_1$
$\mathcal{N}_2$
$\mathcal{N}_k$
$\mathcal{N}_M$

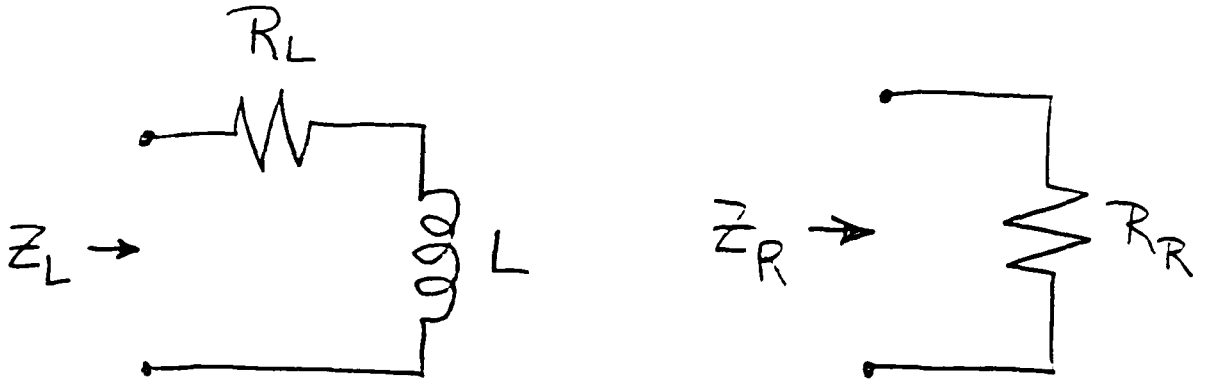

Im $\hat{p}$
$\hat{p}$-plane
λ_{maxAH}
Numerical Range
λ_{minH}
λ_{maxH}
Re $\hat{p}$
λ_{minAH}

3a

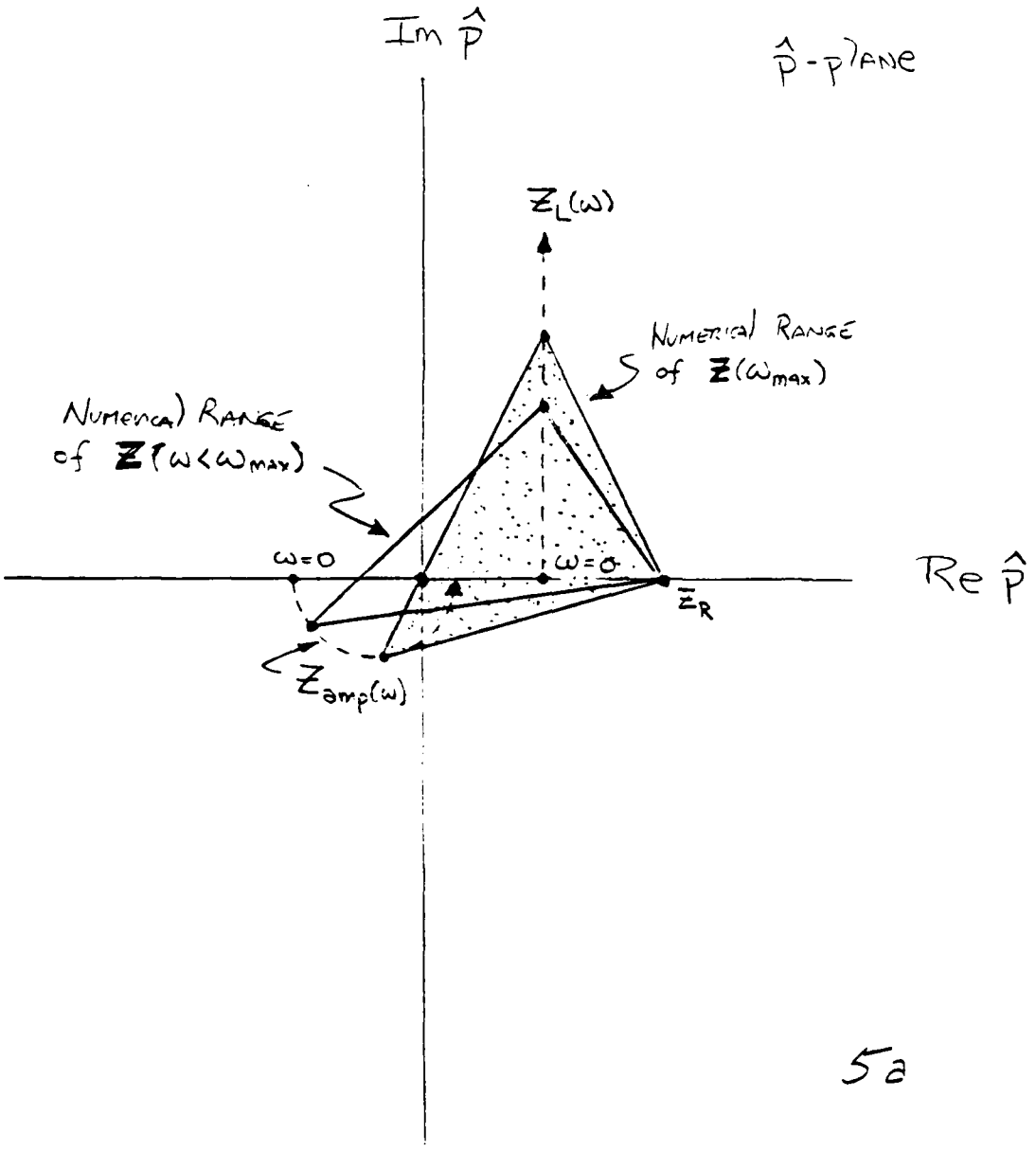

$\mathrm{Im}(\hat{p}e^{j5\pi/4})$
$\mathrm{Re}(\hat{p}e^{j5\pi/4})$



4a



4b


Im $\hat{p}$
$\hat{p}$-PLANE
$Z_L(\omega)$
Numerical Range of $\mathbf{Z}(\omega_{max})$
Numerical Range of $\mathbf{Z}(\omega<\omega_{max})$
$\omega=0$
$\omega=0$
Z_R
$Z_{amp}(\omega)$
Re $\hat{p}$

5a

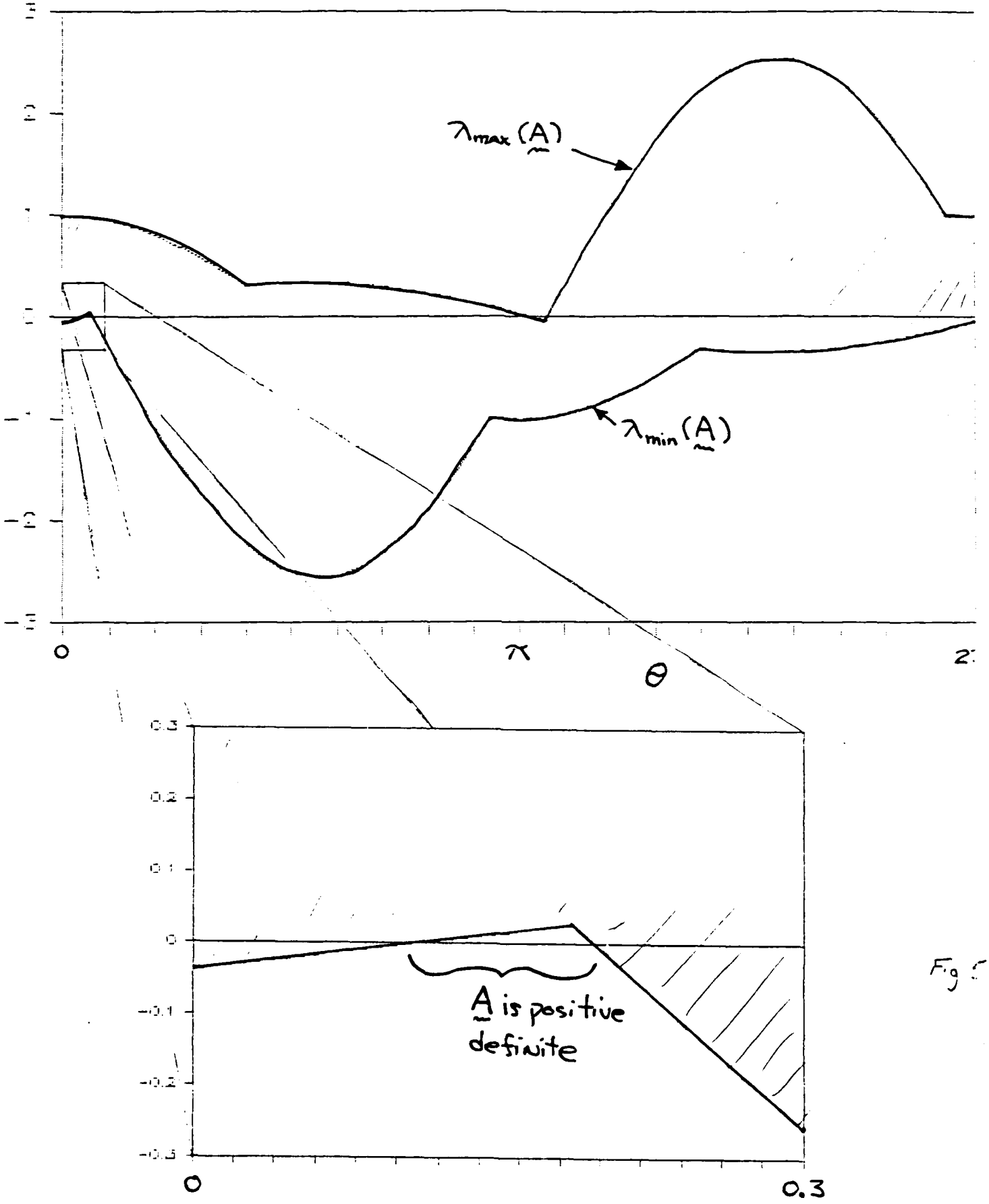

$\lambda_{max}(\underset{\sim}{A})$
$\lambda_{min}(\underset{\sim}{A})$
0
π
θ
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.3
$\underset{\sim}{A}$ is positive definite

Fig 5

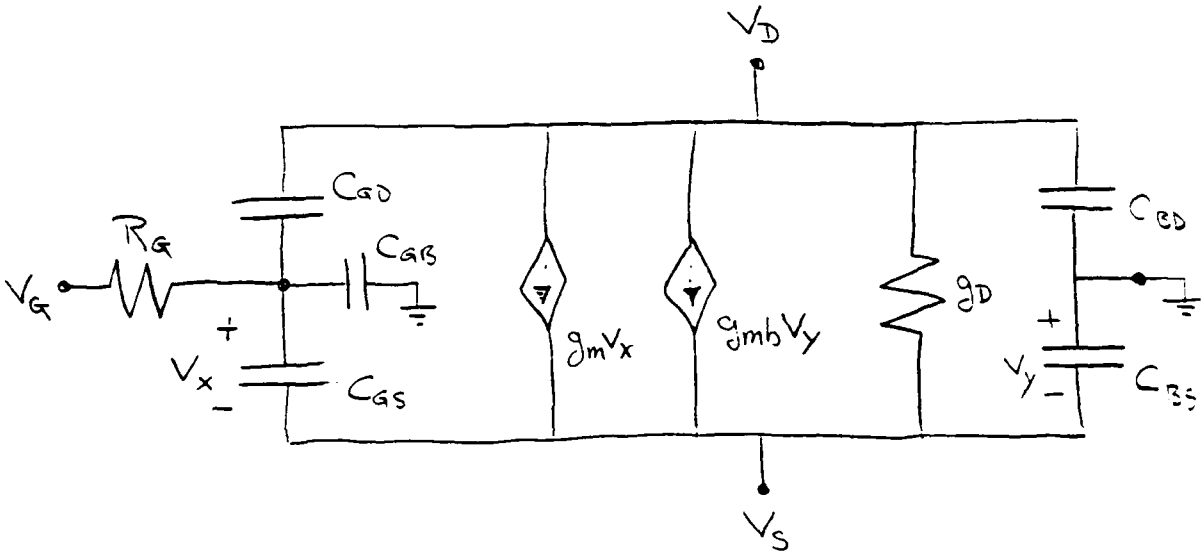


FIG. 6

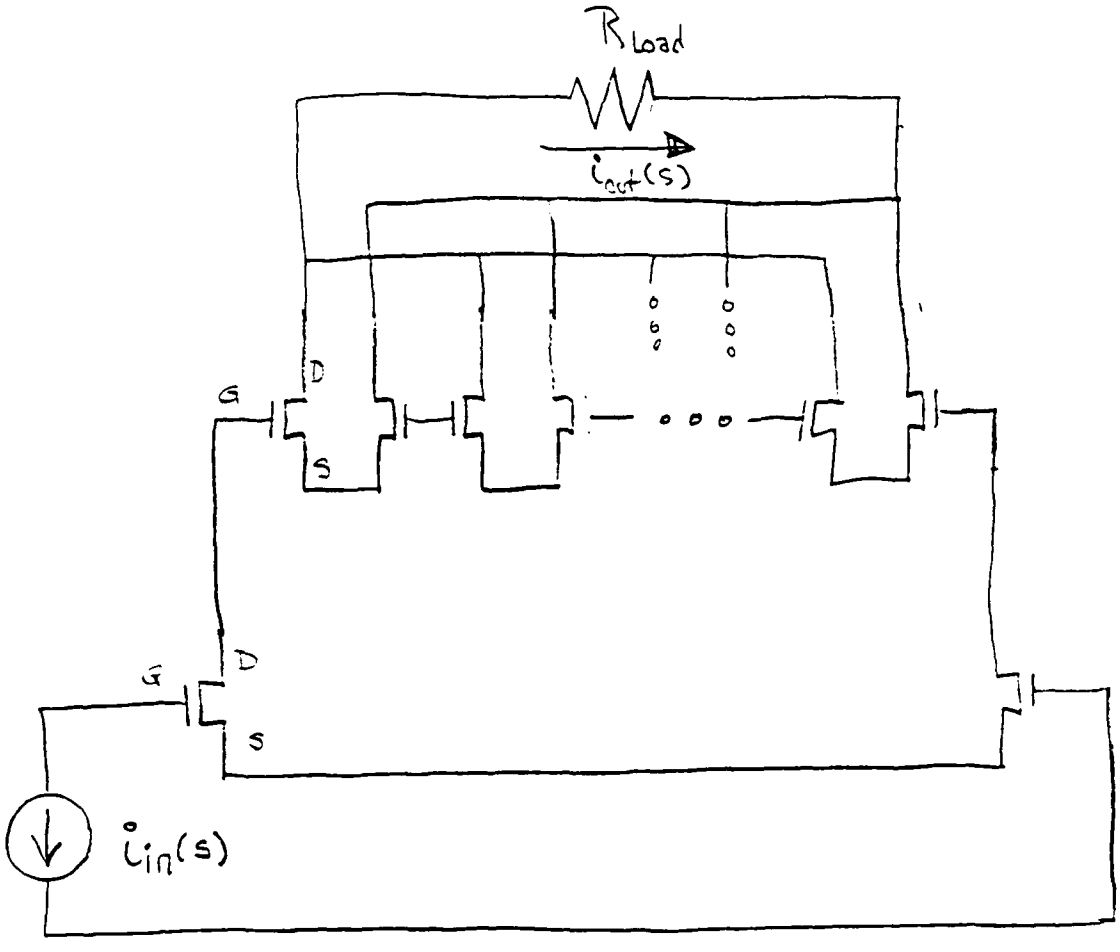


FIG 7

VLSI Memo No. 87-360
January 1987

RELIC: A RELIABILITY SIMULATOR FOR INTEGRATED CIRCUITS

Teresa S. Hohol and Lance A. Glasser

Abstract

Many of the failure mechanisms which cause reliability problems in VLSI chips can be influenced or avoided in the circuit design phase. RELIC is a reliability simulator developed to analyze and predict the stress and wear on MOS VLSI chips due to such mechanisms. RELIC uses a simple methodology for abstracting the idea of the stress from any particular failure mechanism, thus allowing analyses of many different failure mechanisms. There are currently three failure mechanisms analyzed by RELIC: metal migration, hot-electron trapping, and time-dependent dielectric breakdown (TDDB).

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Acknowledgements

Published in Proc. International Conference on Computer-Aided Design, Santa Clara, CA, pp. 517-520, November 11-13, 1986. This paper will appear (translated into Japanese) in Nikkei Microdevices. This work was supported in part by the Defense Advanced Research Projects Agency under contract no. N00014-80-C-0622.

Author Information

Hohol, current address: Hanscom Air Force Base, MA 01731; Glasser: Department of Electrical Engineering and Computer Science and Research Laboratory of Electronics, MIT, Room 36-880, Cambridge, MA 02139, (617) 253-4677.

RELIC: A RELIABILITY SIMULATOR FOR INTEGRATED CIRCUITS

Teresa S. Hohol
Lance A. Glasser

Department of Electrical Engineering and Computer Science
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA. 02139

ABSTRACT *

Many of the failure mechanisms which cause reliability problems in VLSI chips can be influenced or avoided in the circuit design phase. RELIC is a reliability simulator developed to analyze and predict the stress and wear on MOS VLSI chips due to such mechanisms. RELIC uses a simple methodology for abstracting the idea of the stress from any particular failure mechanism, thus allowing analyses of many different failure mechanisms. There are currently three failure mechanisms analyzed by RELIC: metal migration, hot electron trapping, and time dependent dielectric breakdown (TDDB).

1. Introduction

RELIC is a reliability simulator developed to analyze the stress and wear on MOS VLSI chips. RELIC is designed to help the circuit designer develop competitive and reliable chips with minimal extra effort. By using RELIC, chip designers can design, not only for worst-case speed and power [Glasser 85], but also for worst-case reliability. RELIC is not built around any particular failure phenomena or model; rather, RELIC exists as a system in which many failure mechanisms can be simulated and new models easily implemented.

RELIC simulates those failure mechanisms which are under the control, or at least the influence, of the circuit designer. For such failure mechanisms, there exist models which relate the median time to failure (MTTF) of the device to the operating voltages and currents of the circuit and the actual layout of the chip. RELIC employs a circuit simulator so that the voltages and currents used in stress calculations are worst-case operating waveforms and not just the maximum voltages or currents.

The idea of reliability simulation is not a new one and software tools have been proposed to check circuit designs for certain reliability hazards, such as metal migration or hot electrons. A simulator to determine metal migration was described by Kokkonen [Kokkonen 84]. Substrate current circuit simulators have been described by Sing [Sing 80] and Sakurai [Sakurai 85], from which predictions about hot electron trapping can be made. Another hot electron simulator has been proposed by C. T. Sah [Iversen 86] who is currently developing a hot electron model from a combination of theory and experimental results.

While the simulation of failure mechanisms has been proposed previously, there are two major differences between earlier simulators and RELIC. First, RELIC calculates stress and wear based on actual dynamic voltages and currents determined by circuit simulation, and not on the basis of worst-case static operating conditions. In addition, RELIC is the first reliability simulator to run failure tests for several mechanisms. RELIC provides a structure in which existing models may be easily changed, and new models implemented with moderate effort. This is accomplished by the use of a simple methodology for handling the stress and wear caused by many different failure mechanisms.

2. Stress and Wear

One of the unifying concepts used in RELIC is that when a device accumulates a certain amount of stress over time (wear), it fails. This stress may be in the form of trapped charge in the gate oxide causing breakdown (TDDB), or in the form of a transistor threshold shift causing circuit malfunction. Regardless of which particular failure phenomena is being testing, we can abstract the idea of stress, and determine the MTTF of the device on the basis of stress and circuit configuration alone.

RELIC predicts the reliability of the circuit by first calculating the wear which each circuit device experiences over one normal cycle of operation. We define a normal cycle of operation to be the time it takes to run the circuitry through one routine of whatever it does. The wear w on each device is calculated as the integral of stress s over time.

$$w(t) \equiv \int_0^t s(t')dt'. \qquad (1)$$

We assume a deterministic point of view which says that the amount of wear a device can stand until it fails is the critical value of wear $w(T_{fail})$. $w(T_{fail})$ is a random variable with a mean and variance which must be determined from tests on fabricated devices. The critical value of wear may also depend on how and where a device is used in a circuit. For example, the amount of hot electron stress a device can endure without failure depends on its position and function in the overall circuit.

Once we know the critical value of wear (distribution) and the stress rate, we can find the time-to-failure, T_{fail}, where

$$T_{fail} = \frac{w(T_{fail})}{\langle s \rangle}, \qquad (2)$$

* Support for this research was provided by the Defense Advanced Research Projects Agency of the Department of Defense under contract number N00014-80-C-0622.

and where the average stress is

$$\langle s \rangle = \frac{1}{T_{\text{fail}}} \int_0^{T_{\text{fail}}} s(t)dt = \frac{w(T_{\text{fail}})}{T_{\text{fail}}}. \quad (3)$$

Note that we are making the assumption that the stress rate, as determined for one normal cycle, will remain constant for the life of the circuit.

Another important assumption here is that the stress, s, is of the form

$$s(t) = A(t)e^{bE} \quad (4)$$

where $A(t)$, b, and E, are quantities whose form varies for each failure model. E is assumed to be approximately constant over time. These assumptions for s are valid because the phenomena which we are considering all have exponential character in energy space. (See example of metal migration model later.) Given the fact that stress s depends exponentially on E, if E is a normal random variable then s is a log-normal variable.

Substituting (3) and (4) into (2), and taking the log of both sides, we then have

$$\ln T_{\text{fail}} = \ln w(T_{\text{fail}}) - \ln\left(\frac{1}{T_{\text{fail}}} \int_0^{T_{\text{fail}}} A(t)e^{bE}dt\right) \quad (5)$$

which reduces to

$$\ln T_{\text{fail}} = \ln w(T_{\text{fail}}) - \ln(A) - bE. \quad (6)$$

Therefore, because E was a normal random variable, and s was consequently log-normal, then T_{fail}, which depends inversely on s also has a log-normal probability distribution. RELIC assumes log-normal distributions for all failure mechanisms.

This generalized model of stress is applicable for all of the failure models which are currently implemented in RELIC. A, b, and E can be determined from model and circuit parameters and circuit simulation. The critical wear value $w(T_{\text{fail}})$ must be determined experimentally and, as mentioned earlier, could depend on a device's location and function in the circuit.

Once RELIC has computed the time-to-failure distribution for each device, it then combines the distributions for each device to obtain the total failure distribution for the entire circuit. Using the assumption that the failure of one device has no effect on the probability of failure of any other device, we then treat the failure probabilities as being independent. Given that, then the probability of system failure P_{sysfail} is $P_{\text{sysfail}} = 1 - P_{\text{syswork}}$, where P_{syswork} is the probability that the system has not failed. Assume that one device failure is enough to cause the system to fail. Given an independent system, P_{syswork} is equal to the product of the probabilities that each device is working. Thus, P_{sysfail} is found to be

$$P_{\text{sysfail}} = 1 - \prod_i (1 - P_{\text{fail}_i}). \quad (7)$$

In addition, if all of the individual probabilities for device failure are small, and the cross products of (7) even smaller, we may then use what is known as the *Rare Event Approximation*. Using this approximation, the probability of system failure becomes the linear sum of the individual failure probabilities for each device. That is,

$$P_{\text{sysfail}} \leq \sum_i P_{\text{fail}_i}. \quad (8)$$

Note that this only works if there are a small number of devices n_{dev} such that $n_{\text{dev}} \ll \frac{1}{P_{\text{fail}_i}}$.

3. System Structure and Implementation

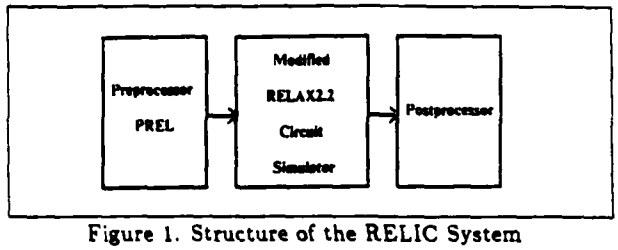


Figure 1. Structure of the RELIC System

The RELIC simulator consists of three parts: a preprocessor, a modified circuit simulator, and a post processor, as illustrated in Fig 1. We use the circuit simulator both to simulate the circuit to determine voltage and current waveforms and to calculate device stress based on these conditions. The simulator in this implementation is RELAX2.2 from the University of California at Berkeley [White 85]. What we have done, basically, is to introduce into RELAX some new device models, one for each failure mechanism we are simulating.

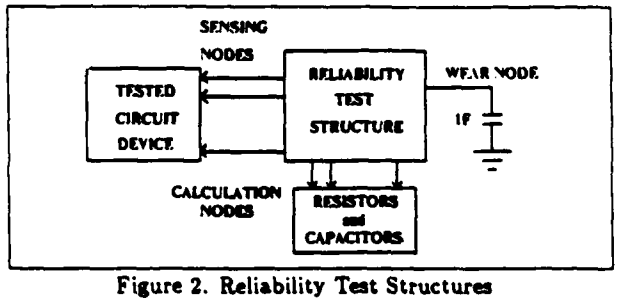


Figure 2. Reliability Test Structures

These new models, which I shall refer to as *reliability test structures* are connected to the circuit nodes of the device undergoing testing to measure its voltages and currents, and from these operating conditions, along with the device size and processing parameters, calculate the instantaneous stress on that device [Fig 2]. Besides sensing the circuit operating conditions, test structures also employ circuit nodes connected to various configurations

of resistors and capacitors to calculate intermediate quantities. A final node, the *wear node*, is used to output the accumulated stress. This node is connected to a grounded capacitor and input a current proportional to the instantaneous stress, so that the voltage across this node represents the total wear incurred.

The RELIC preprocessor, PREL, takes an input file which describes the circuit according to RELAX syntax. This input file must also contain some RELIC commands indicating which devices to test, and for which failure mechanisms. PREL modifies this file to include the appropriate reliability test structures for each device undergoing testing. PREL also instructs the circuit simulator to output the voltage waveform for the new *wear node*. This new circuit configuration is then simulated by the circuit simulator for one normal cycle of operation and the wear incurred for this cycle is output. Finally, the RELIC postprocessor must compare this device wear information with critical failure wear data to determine the time-to-failure of individual devices and the MTTF for the entire circuit. The postprocessor is currently under development at the time of this writing.

4. Failure Models

The current implementation of RELIC contains models for metal migration Kokkonen 84 , hot electron trapping Hsu 82 , and time dependent dielectric breakdown Chen 85 . Presented here are the salient features of the metal migration model, which incorporates the effects of IR heating, thermal capacitance, and thermal resistance. The wire stress depends on the wire temperature and current waveforms.

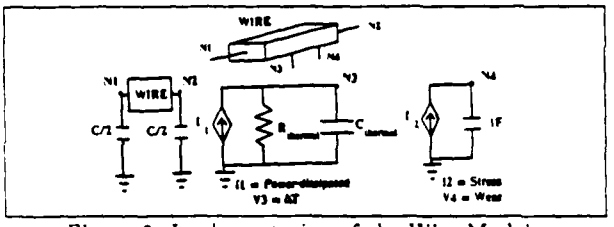


Figure 3. Implementation of the Wire Model

The wire model is a 4-node reliability test structure. Two of the nodes, *N1* and *N2*, represent the ends of the wire and are connected to the circuit nodes on either end of the wire. (This feature is not well-supported by layout extractors, which consider a wire to be one node in the circuit; consequently, the locations of all wires the user wishes to simulate must be identified in the PREL input file.) RELIC models the electrical behavior of the wire as a pi model, having the wire's resistance between *N1* and *N2* and half the wire capacitance from each of these nodes to ground [Fig 3].

The wire model also employs a node, *N3*, for use in calculating an intermediate quantity. The stress on the wire is dependent on the wire temperature, which is a function of the thermal capacitance and resistance. The thermal equivalent RC circuit is also shown in [Fig 3], where the voltage on *N3* represents the change in temperature of the wire.

The final node of the wire model is *N4*, which is the *wear node*. The current source, *S2*, is proportional to the instantaneous stress calculated by the metal migration equations and, therefore, the voltage on *N4* is proportional to the amount of metal migration stress on the wire.

The stress on the wire due to metal migration is roughly expressed as

$$s(t) = AJ^2(t)e^{(\frac{E_a}{kT})} \qquad (9)$$

where A is related to the wire's physical size, J is the current density through the wire, E_a is the activation energy, k is Boltzman's constant, and T is the wire temperature. Note that this is consistent with our assumption earlier that the stress in our models was exponential in energy space. This is also true in the hot electron and TDDB models. Details on all of the equations and parameters used in RELIC can be found in Hohol 86 .

The hot electron and TDDB models are implemented in basically the same way as the metal migration model. Both of these reliability test structures have sensing nodes which connect in parallel to the source, gate, drain, and bulk nodes of the transistor being tested. Because these test structures are inserted in parallel with the circuit device being tested, the user does not have to specify additional nodes in the circuit (unlike the wire model, which is inserted serially and requires two nodes instead of one).

5. Results: An Example

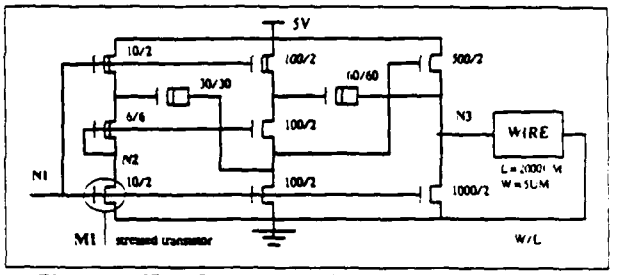


Figure 4. IBM Bootstrapped Superbuffer, Stressed

RELIC was used to analyze one version of an IBM Bootstrapped Superbuffer, shown in Fig 4. This circuit uses two stages of bootstrapping to in order to drive a large capacitive load. However, the result of this double bootstrapping is a large amount of hot electron and TDDB stress on transistor *M1*.

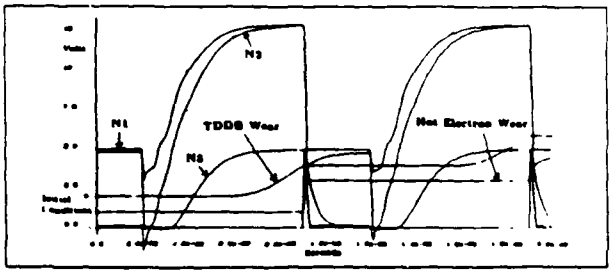


Figure 5. Hot Electron and TDDB Wear on *M1*

When the input *N1* to the superbuffer is low and the bootstrap action has been completed, the source and gate of transistor *M1* are at 0v and the drain *N2* is around

12.5 v. This large voltage differential across the oxide between the gate *N1* and the drain *N2* creates large amounts of TDDB stress and wear (See Fig 5). When the input *N1* to the superbuffer rises again, transistor *M1* turns on, with an initial drain-to-source voltage of 12.5 v. This large saturation voltage generates hot electron stress and wear on the gate oxide of *M1* (also in Fig 5).

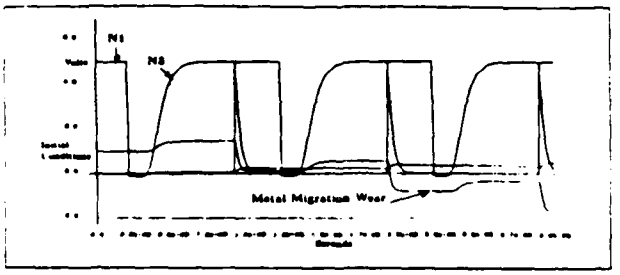


Figure 6. Metal Migration Wear on Output Wire

In order to test RELIC on a metal migration simulation, I added a 2000 μm long wire with a width of 5 μm to the output node *N9* of the superbuffer. As this wire was charged and discharged, the currents through the metal were found to cause metal migration wear. This wear is shown on the plot in Fig 6. Note that the metal migration wear increases and decreases as the current in the wire changes direction, but that the net wear appears to be in the negative direction. This does not mean that there is negative wear on the wire, but that the wear caused by the wire discharging is greater than the wear endured during charging. This is because the wire discharges faster than it charges, and consequently the currents during this period are greater in magnitude and cause more stress according to our simple model.

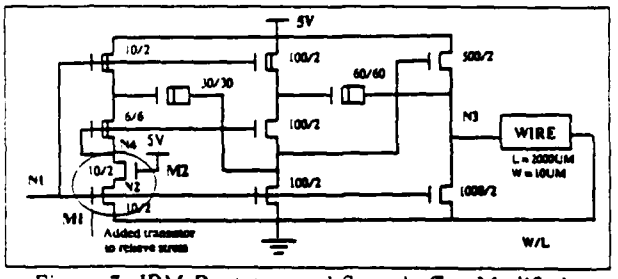


Figure 7. IBM Bootstrapped Superbuffer, Modified

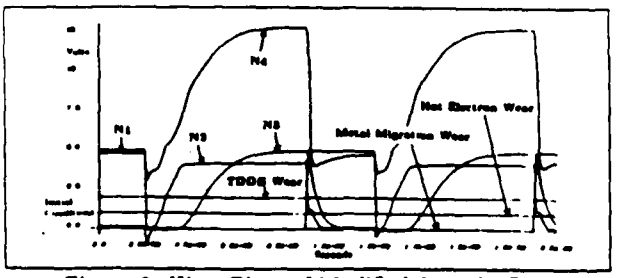


Figure 8. Wear Plots of Modified Superbuffer

An alternative IBM Bootstrapped Superbuffer is shown in Fig 7. This circuit has an added transistor *M2*, which protects the drain of *M1* from voltages higher than $V_{dd} - V_{TH}$ (See Fig 8). The 12.5 v voltage drop is now spread across two transistors instead of one. Although the drain of *M2* is still at 12.5 v, the gate of that transistor is at 5 v which reduces to voltage across the gate oxide to 7.5 v, which does not register any TDDB stress for the oxide thickness of 125 Å. Similarly, the approximately 8 v voltage drop from the drain to the source of *M2* does not register any hot electron effects (See Fig 8). In order to reduce the metal migration stress, I increased the wire width to 10 μm. This causes the current density in the wire to decrease, and consequently, no metal migration wear is measured (also in Fig 8).

6. Conclusions

In this paper we have presented RELIC, a reliability simulator for determining stress and wear on integrated circuits. RELIC employs a unifying strategy for abstracting stress from individual failure mechanisms, and therefore allows for the analysis of many different failure mechanisms. The heart of the RELIC simulator is a circuit simulator, employed both to determine instantaneous voltage and current waveforms and to carry out the actual equations for calculating stress due to failure mechanisms. The analyses carried out by RELIC on two versions of the IBM Bootstrapped Superbuffer show how this tool may be used by circuit designers to both identify problems and verify their correction.

References

[Chen 85] I.C. Chen, S.E. Holland, and C. Hu, "Electrical Breakdown in Thin Gate and Tunneling Oxides," *IEEE Trans. Elec. Dev.* 32: 413-422, 1985.

[Glasser 85] L. A Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Reading, MA: Addison-Wesley, 1985.

[Hohol 86] T. S. Hohol, "RELIC: A Reliability Simulator for Integrated Circuits," S.M. Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1986.

[Hsu 82] F. C. Hsu, P. K. Ko, S. Tam, C. Hu, and R. S. Muller, "An Analytical Breakdown Model for Short-Channel MOSFET's," *IEEE Trans. Elec. Dev.* 29: 1735-1740, 1982.

[Iversen 86] W. R. Iversen, "Model May Help Solve Chip-Reliability Problem," *Electronics* pp. 20-21, March 24, 1986

[Kokkonen 84] K. Kokkonen, "The Interaction of Physics and CAD in VLSI CMOS Design," talk given at MIT Nov 1984.

[Sakurai 84] T. Sakurai, M. Kakumu, T. Iizuka, "Hot-Carrier Suppressed VLSI with Submicron Geometry," *IEEE/ISSCC* 272-273, 1985.

[Sing 80] Y. W. Sing and B. Sudlow, "Modeling and VLSI Design Constraints of Substrate Current," *IEEE/IEDM* 732-734, 1980.

[White 85] J. K. White,and A. Sangiovanni-Vincentelli, "RELAX2.1 - A Waveform Relaxation Based Circuit Simulation Program," *Proc. 1984 Int. Custom Integrated Circuits Conference*, Rochester, N.Y. June, 1984.

Computer-Aided Circuit Reliability Work at M.I.T.

Lance A. Glasser

January 6, 1987

The first formal M.I.T. course on transistors was taught in 1953. Dispite this early start, for most of the 70's the Institute maintained only a small integrated circuit research effort. In 1977 M.I.T. made a strategic decision to reemphasize microsystem technology and since that time the size of this program has increased dramatically, with new faculty being hired each year. The program is now large and vibrant, spanning research areas from nanometer structures to multiprocessor architectures. One of the many areas which has received recent attention is electrical issues in large digital machines, where a "large" machine is one whose physical dimensions are big compared to the distance a bit spans as it speeds across the machine. Within this area, five critical topics are being addressed: I/O, synchronization (e.g., clocking), power, noise, and reliability. It is this last topic which is addressed in the accompanying article.

RELIC is our first attempt to design a simulation program which enables the engineer to design high-performance circuits not only for worst-case speed, power, and noise margin, but also worst-case reliability. Our program is the first to support the reliability simulation of a circuit stressed by several dynamic reliability hazards.

As process and device technologies advance, the constraints that must be dealt with continually become more complex and difficult. Nevertheless, this complex constraint space is today reflected in the circuit domain as an orthogonal set of relatively simple rules. We do not believe that this simplicity can be maintained. In the future, product competitiveness will be determined, in part, by the ability of the circuit designer to design systems which simultaneously extract the maximum performance from critical devices while avoiding the edge of complex-shaped low-reliability regimes. This will be possible only with the use of high-quality reliability models and computer-aided design tools. RELIC is a demonstration of the sort of low-level design tools which will be necessary. (It is also worthwhile to ask about higher-level tools which aid reliability design.)

One of the limitations of RELIC is that reliability models are not sufficiently developed to predict the failure rate of a chip, even given exact process and circuit models. Nevertheless, it should be possible to compare two circuits for relative reliability and thereby guide the design of high-reliability parts. There is a second, less obvious, application. One commonly used technique for improving the reliability of a part is to do an accelerated burn-in to remove the weak devices, those which contribute to infant mortality. It is not always clear, however, how to accelerate the stress on a part because, though one may raise the power-supply voltage from 5 to 7 V, the voltages internal to the chip need not follow. (Consider voltages controlled by current mirrors or charge pumps.) One therefore needs to *design for stressability.* For high reliability applications one must be able to design a circuit so that accelerated burn-in can be accomplished. RELIC is suited to this task.

RELIC is a first-generation program. It is an experimental program written by Miss Terry Hohol on an experimental program (RELAX 2.2). It is therefore not surprising that while the program is operational, it is neither robust nor user friendly. But we have learned many things from RELIC. A second-generation program, now under development, will improve upon RELIC in five ways: (1) it will be based on a more solid circuit simulation program; (2) the models will be improved based on our better understanding of the literature; (3) the control structure will be modified so that it is easy to see the long-term effects of transconductance, threshold, and leakage degradation on circuit performance; (4) a post-processor will be added which predicts failure rates and cumulative percent fails in terms of the "wear" simulation variables. This means that one must model the MTTF and σ of as many as three statistical populations (main, freak, infant); and (5) we intend to make the second-generation program more robust and hence usable by the community.

Assuming that we can accomplish these tasks, a simulation program is still only as good as the models. Improved models are desperately needed. Even after all these years of research, metal migration is still not well understood. For instance, one can find in the literature papers that predict that pulsed operation is better, and worse, than steady-state operation. Hot carrier models are in reasonable shape for dc excitation but, again, the effects of trapping and de-trapping time constants on dynamic stress is unclear. Time-dependent dielectric breakdown is not well modeled even under dc excitation—from electric field data there appear to be at least two competing mechanisms—and pulsed dynamics and interactions with hot-carrier stressing are generally mysterious. Quality programs to do dynamic reliability simulation will soon exist. It is hoped that the reliability physics community will be able to meet the enormous challenge of quantifying the dynamics of device failure under stress so that these programs can accurately predict system reliability.

Efficient Parallel Algorithms for $(\Delta+1)$-Coloring and Maximal Independent Set Problems*

Andrew V. Goldberg
Serge A. Plotkin

Laboratory for Computer Science
MIT
Cambridge MA 02139

January 1987

*This work was supported in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contracts N00014-80-C-0622 and N00014-75-C-0661. The first author is supported in part by Fannie and John Hertz Foundation Fellowship.

Abstract

We describe an efficient technique for breaking symmetry in parallel. The technique works especially well on rooted trees and on graphs with a small maximum degree. In particular, we can find a maximal independent set on a constant-degree graph in $O(\lg^* n)$ time on an EREW PRAM using a linear number of processors. We show how to apply this technique to construct more efficient parallel algorithms for several problems, including coloring of planar graphs and $(\Delta+1)$-coloring of constant-degree graphs. We also prove lower bounds for two related problems.

1 Introduction

Some problems for which trivial sequential algorithms exist appear to be much harder to solve in a parallel framework. Therefore, new methods are needed for design of efficient parallel algorithms. A known example of a problem with a trivial sequential algorithm which is hard to solve in parallel, is the problem of finding a maximal independent set in a graph [18]. This problem was shown to be in the class NC by Karp and Wigderson [12]. A simple randomized algorithm for the problem is due to Luby [15]. Recently M. Goldberg and Spencer [9] gave a deterministic algorithm for the problem that runs in polylogarithmic time using a linear number of processors.

The study of the maximal independent set problem shows the importance of techniques for breaking symmetry in parallel. The symmetry-breaking comes up in many other parallel algorithms as well. In many cases, however, it is enough to be able to break symmetry in special kinds of graphs. The performance of the resulting algorithm improves if we can solve the special case of symmetry-breaking more efficiently.

In this paper we present a technique for breaking symmetry. In particular, we give an $O(\lg^* n)$ time algorithm to 3-color a rooted tree. This techniques can be viewed as a generalization of the deterministic coin-flipping technique of Cole and Vishkin [5]. To show the usefulness of our technique, we present the following algorithms. All of the presented algorithms use a linear number of processors.

- For graphs whose maximum degree is a constant Δ, we give an $O(\Delta \lg \Delta \lg^* n)$ algorithm for $(\Delta+1)$-coloring and for finding a maximal independent set on an EREW PRAM.
- We give an algorithm to 7-color a planar graph. This algorithm, and the maximal independent set (for planar graphs) algorithm based on it, run in $O(\lg n \lg^* n)$ time on a CRCW PRAM and in $O(\lg^2 n)$ time on an EREW PRAM. We also give an $O(\lg^3 n \lg^* n)$ CRCW algorithm to 5-color a planar graph.
- We give a $O(\lg n \lg^* n)$ algorithm for finding a maximal matching in a planar graph on a CRCW PRAM.
- For general graphs we give an $O(\Delta^2 \lg n)$ algorithm for $(\Delta+1)$-coloring and for finding a maximal independent set on EREW PRAM.

The above stated results improve the running time and processor bounds for the respective problems. The fastest previously known algorithm for $(\Delta+1)$-coloring [15], in the case of constant-degree graphs, runs in $O(\lg n)$ time, and the deterministic version of this algorithm requires n^3 processors. The 5-coloring algorithm for planar graphs, due to Boyar and Karloff, [4] runs in $O(\lg^3 n)$ time, and the deterministic version of this algorithm requires n^3 processors. The $O(\lg^3 n)$ running time of the maximal matching algorithm due to Israeli and Shiloach [10] can be reduced to $O(\lg^2 n)$ in the restricted case of planar graphs, but our algorithm is faster.

Although in this paper we have limited ourselves to the application of our techniques for the design of of parallel algorithms for the PRAM model of computation, the same techniques can be applied in a distributed model of computation [1,7]. Moreover, the $O(\lg^* n)$ lower bound, given by Linial [14] for the maximal independent set problem on a chain in the distributed model, shows that our symmetry-breaking technique is optimal in this model.

The fact that a rooted tree can be 3-colored in $O(\lg^* n)$ time raises the question whether a rooted tree can be 2-colored within the same time complexity. We answer this question by giving an $\Omega(\lg n / \lg\lg n)$ lower bound for 2-coloring of a rooted tree.

We also present an $\Omega(\lg n/\lg\lg n)$ lower bound for finding a maximal independent set in a general graph, thus answering the question posed by Luby [15].

Some of the results presented here were obtained independently by Shannon [16].

2 Definitions and Notation

This sections describes the assumptions about the computational model, and introduces the notation used throughout the paper. In this paper we use n to denote the number of vertices and m to denote the number of edges in a graph. We use Δ to denote the maximum degree of the graph.

Given a graph $G = (V, E)$, we say that a subset of nodes $I \in V$ is *independent* if no two nodes in I are adjacent. A *coloring* of a graph G is an assignment $C : V \rightarrow N$ of positive integers (colors) to nodes of the graph. A coloring is *valid* if no two adjacent nodes have the same color. The i^{th} bit in the color of a node v is denoted by $C_v(i)$. A subset of edges $M \in E$ is a matching if any two distinct edges in M have no nodes in common.

The following problems are discussed in the paper:

- The vertex-coloring (VC) problem: find a valid coloring of a given graph that uses at most $\Delta+1$ colors.
- The maximal independent set (MIS) problem: find a maximal independent set of vertices in a given graph.
- The maximal matching (MM) problem: find a maximal matching in a given graph.

We make a distinction between *unrooted* and *rooted* trees. In a rooted tree, each nonroot node knows which of its neighbors is its parent.

The following notation is used:

$$\begin{aligned} \lg x &= \log_2 x \\ \lg^{(1)} x &= \lg x \\ \lg^{(i)} x &= \lg\lg^{(i-1)} x \\ \lg^* x &= \min\{i \mid \lg^{(i)} x \leq 2\} \end{aligned}$$

We assume a PRAM model of computation where each processor is capable of executing simple word and bit operations. The word width is assumed to be $O(\lg n)$. The word operations we use include bit-wise boolean operations, integer comparisons, and unary-to-binary conversion. In addition, we assume that each processor has a unique *identification number* $O(\lg n)$ bits wide, which we denote by PE-ID. We use exclusive-read, exclusive-write (EREW) PRAM, concurrent-read, exclusive-write (CREW) PRAM, and concurrent-read, concurrent-write (CRCW) PRAM as appropriate. All lower bounds are proven for a CRCW PRAM with a polynomial number of processors.

3 Coloring Rooted Trees

This section describes an $O(\lg^* n)$ time algorithm for 3-coloring rooted trees. First we describe an $O(\lg^* n)$ time algorithm for 6-coloring rooted trees. Then we show how to transform a 6-coloring of a rooted tree into 3-coloring in constant time.

The procedure *6-Color-Rooted-Tree* is shown in Figure 1. This procedure accepts a rooted tree $T = (V, E)$ and 6-colors it in time $O(\lg^* n)$. Starting from the valid coloring given by the processor ID's, the procedure iteratively reduces the number of bits in the color descriptions by recoloring each non-root node v with the color obtained by concatenating the index of a bit in which C_v differs from $C_{father(v)}$ and the value of this bit. The root r appends $C_r[0]$ to 0.

Theorem 1 *The algorithm* 6-Color-Rooted-Tree *produces a valid 6-coloring of a tree in* $O(\lg^* n)$ *time on a CREW PRAM using* $O(n)$ *processors.*

```
PROCEDURE 6-Color-Rooted-Tree
L ← ⌈lg n⌉
for all v ∈ V in parallel do C_v ← PE-ID(v)   ;;; initial coloring
while L > ⌈lg L + 1⌉ for all v ∈ V in parallel do
      if v is the root
        then do
             i_v ← 0
             b_v ← C_v(0)
             end
        else do
             i_v ← min{i | C_v(i) ≠ C_father(v)(i)}
             b_v ← C_v(i_v)
             end
      C_v ← b_v i_v
end
```

Figure 1: The Coloring Algorithm for Rooted Trees

Proof: First we prove by induction that the coloring computed by the algorithm is valid, and then we prove the upper bound on the execution time.

Assume that the coloring C is valid at the beginning of an iteration, and show that the coloring at the end of the iteration is also valid. Let v and w be two adjacent nodes; without loss of generality assume that v is the father of w. By the algorithm, w chooses some index i such that $C_v(i) \neq C_w(i)$ and v chooses some index j such that $C_v(j) \neq C_{father(v)}(j)$. The new color of w is $\langle i, C_w(i)\rangle$ and the new color of v is $\langle j, C_v(j)\rangle$. If $i \neq j$, the new colors are different and we are done. On the other hand, if $i = j$, then $C_v(i) \neq C_w(i)$ and again the colors are different. Hence, the validity of the coloring is preserved.

Now we show that the algorithm terminates after $O(\lg^* n)$ iterations. Let L_k denote the number of bits in the representation of colors after k iterations. For $k = 1$ we have

$$\begin{aligned} L_1 &= \lceil \lg L \rceil + 1 \\ &\leq 2\lceil \lg L \rceil \end{aligned}$$

if $\lceil \lg L \rceil \geq 1$.

Assume for some k we have $L_{k-1} \geq 2\lceil \lg^{(k-1)} L \rceil$ and $\lceil \lg^{(k)} L \rceil \geq 2$. Then

$$\begin{aligned} L_k &= \lceil \lg L_{k-1} \rceil + 1 \\ &\leq \lceil \lg(2 \lg^{(k-1)} L) \rceil + 1 \\ &\leq 2\lceil \lg^{(k)} L \rceil \end{aligned}$$

Therefore, as long as $\lceil \lg^{(k)} L \rceil \geq 2$,

$$L_k \leq 2\lceil \lg^{(k)} L \rceil.$$

Hence, the number of bits in the representation of colors L_k decreases until, after $O(\lg^* n)$ iterations, $\lceil \lg^{(k)} L \rceil$ becomes 1 and L_k reaches the value of 3 (the solution of $L = \lceil \lg L \rceil + 1$). Another iteration of the algorithm produces a 6-coloring: 3 possible values of the index i_v and 2 possible values of the bit b_v. The algorithm terminates at this point.

We use concurrent-read capability to broadcast the newly computed color C_v to all the sons of v; no concurrent-write capabilities are required. For constant-degree trees the concurrent-read capability is not needed either. ∎

As we have shown, a rooted tree can be 6-colored quickly. A natural question to ask at this point is whether one can use less colors and still stay within the same complexity bounds. The following theorem answers this question.

Theorem 2 *A rooted tree can be 3-colored in $O(\lg^* n)$ CREW PRAM time using $O(n)$ processors.*

Proof: The algorithm *3-Color-Rooted-Tree* presented in Figure 2 starts by using the previously described algorithm to 6-color the tree and then recolors it in 3 colors in constant time.

The algorithm recolors the nodes colored with *bad* colors 3, 4, and 5, into *good* colors 0, 1, 2 as follows. First, each node is recolored in the color of its father, so that any two nodes with the same father have the same color. The root, which

has no father, recolors itself with a color different from its current color. Next, the algorithm removes the color from every node that has a bad color and has a neighbor with a good color. These nodes become uncolored. Every node v that still has a color C_v is recolored in the color $C_v \bmod 3$; this gets rid of the remaining bad colors. Note that this coloring has the nice property that for any node v, all of the sons of v that are colored, must be colored identically.

The resulting coloring is valid, but not all nodes are colored. By the construction, every uncolored node has at least one colored neighbor. Therefore, if there are two nodes v and w, such that $v = father(w)$ and both nodes are uncolored, then $father(v)$ is colored and $sons(w)$ are colored too. The algorithm colors v with a color different from $C_{sons(v)}$ and from $C_{father(v)}$. Such a color always exists because there are 3 different colors to choose from and all the colored sons of v are colored with the same color. Finally, the algorithm colors w with a color different from both C_v and $C_{sons(w)}$. Every step of the *3-Color-Rooted-Tree* algorithm can be executed in constant time except for the first one, in which we color the tree with 6 colors. Hence, the total running time of the algorithm is $O(\lg^* n)$. ∎

Any tree can be 2-colored. In fact, it is easy to 2-color a tree in polylogarithmic time. For example, one can use treefix operations [13] to compute the distance from each node to the root, and color even level nodes with one color and odd level nodes with the other color. It is harder to find a 2-coloring of a rooted tree in parallel, however, than it is to find a 3-coloring of a rooted tree. In section 7 we show a lower bound of $\Omega(\lg n / \lg\lg n)$ on 2-coloring of a directed list by a CRCW PRAM with a polynomial number of processors, which implies the same lower bound for rooted trees.

4 Coloring Constant-Degree Graphs

The method for coloring rooted trees described in the previous section is a generalization of the deterministic coin-flipping technique described in [5]. The method can be generalized even further [8] to color constant-degree graphs in a constant

PROCEDURE *3-Color-Rooted-Tree*
$C \leftarrow$ 6-Color-Rooted-Tree (V, E)
for all $v \in V$, $v \neq$ root **in parallel do**
    $C_v \leftarrow C_{father(v)}$
end
$C_{root} \leftarrow \min\{\ \{0,1,2\} - \{C_{sons(root)}\}\ \}$
$V_1 \leftarrow \{v \mid C_v \leq 2\}$
$V_2 \leftarrow V - V_1$
$V' \leftarrow \{v \mid v \in V_2$ and $\exists (v,w) \in E, w \in V_1\}$;;; *bad*-colored nodes with *good*-colored neighbors
for all $v \in V - V'$ **in parallel do**
    $C_v \leftarrow C_v \bmod 3$
end
for all $v \in V'$ **in parallel do**
    $C_v \leftarrow$ uncolored
end
for all $v \in V'$ **in parallel do**
    if father(v) $\notin V'$
      then do
          $C_v \leftarrow \min\{\ \{0,1,2\} - \{C_{sons(v)}\} - \{C_{father(v)}\}\ \}$
          $V' \leftarrow V' - v$
      end
end
for all $v \in V'$ **in parallel do**
    $C_v \leftarrow \min\{\ \{0,1,2\} - \{C_{sons(v)}\} - \{C_{father(v)}\}\ \}$
end

Figure 2: The 3-coloring Algorithm for Rooted Trees

number of colors. In the generalized algorithm, a current color of a node is replaced by a new color obtained by looking at each neighbor, appending the index of a bit in which the current color of the node is different from the neighbor's color to the value of the bit in the node color, and concatenating the resulting strings. This algorithm runs in $O(\lg^* n)$ time, but the number of colors, although constant, is exponential in the degree of the graph.

In this section we show how to use the procedure *3-Color-Rooted-Tree* described in the previous section to color a constant-degree graph with $(\Delta+1)$ colors, where Δ is the maximum degree of the graph.

First, we describe how to find in constant time a forest in a given graph such that each node with nonzero degree in the graph has nonzero degree in the forest. The removal of the edges of the forest decreases the maximum degree of the remaining graph (unless the maximum degree of the graph is zero). We shall use this property later use to decompose the edges into Δ sets, each set inducing a forest on the nodes of the graph. The procedure *Find-Forest* (see Figure 3) constructs such a forest.

The procedure has two steps. In the first step each node compares the ID's of its neighbors with its own ID. A node that does not have the maximum processor ID among its neighbors chooses an edge that connects it to the neighbor with the largest processor ID. The graph induced by the chosen edges is a forest (the graph has no cycles) and the nodes with the highest processor IDs among their neighbors – local maximums – are roots of the forest. In the second step each root with no sons chooses an edge that connects it to one of its neighbors. The roots are local maximums and are therefore independent. Hence, no new cycles are introduced into the graph induced by the chosen edges.

The algorithm *Color-Constant-Degree-Graph* that colors constant-degree graph with $(\Delta+1)$ colors is presented in Figure 4. The algorithm consists of two phases. In the first phase we iteratively call the *Find-Forest* procedure, each time removing the edges of the constructed forest. This phase continues until no edges remain, At which point we color all the nodes with one color.

In the second phase we iteratively return the edges of the forests into the graph,

```
PROCEDURE Find-Forest(V,E)
E' ← ∅
R ← ∅
for all v ∈ V in parallel do   ;;; construct the forest – the first step
    if PE-ID(v) is not a local maximum
      then do
            e_v ← (v, w) s.t. (v, w) ∈ E and PE-ID(w) = max{PE-ID(u)|(v, u) ∈ E}
            E' ← E' ∪ e_v
      end
      else do
            R ← R ∪ v
      end
end
for all v ∈ R in parallel do   ;;; get rid of zero-depth trees – the second step
    if ∄(v, w) ∈ E' and ∃(v, w') ∈ E
      then do
            E' ← E' ∪ (v, w')
      end
end
return (E') ;;; the edges of the forest
```

Figure 3: The Spanning Forest Algorithm

```
PROCEDURE Color-Constant-Degree-Graph
E' ← E
i ← 0
while E' ≠ ∅ do   ;;; the first phase
      E_i ← Find-Forest(V, E')
      E' ← E' − E_i
      i ← i + 1
end
for all v ∈ V in parallel do   ;;; initial coloring
    C(v) ← 1
end
for i ← i − 1 to 0 do   ;;; the second phase
    C' ← 3-Color-Rooted-Tree (V, E_i)
    E' ← E' + E_i
    for k ← 1 to 3 do
        for j ← 1 to Δ+1 do
            V' ← V
            for all v ∈ V' in parallel do
                if C(v) = j and C'(v) = k
                  then do
                        C(v) ← max {{1, 2, ...Δ+1} − {C(w) | (v, w) ∈ E'}}
                        V' ← V' − v
                  end
            end
        end
    end
end
```

Figure 4: The Recoloring Algorithm for Constant Degree Graphs

each time recoloring the nodes to maintain a consistent coloring. At the beginning of each iteration of this phase, the edges of the current forest (E') are added, making the existing $(\Delta+1)$-coloring inconsistent. This forest is colored with 3 colors using the *3-Color-Rooted-Tree* procedure. Now, each node has two colors – one from the coloring at the previous iteration and one from the coloring of the forest. The pairs of colors form a valid $3(\Delta+1)$-coloring of the graph. The iteration finishes by enumerating the color classes, recoloring each node of the current color with a color from $\{0,\ldots,\Delta\}$ that is different from the colors of its neighbors (note that we can recolor all the nodes of the same color in parallel because they are independent).

Theorem 3 *The algorithm* Color-Constant-Degree-Graph *runs in* $O(\Delta \lg \Delta(\Delta + \lg^* n))$ *time and colors the graph with* $(\Delta+1)$ *colors.*

Proof: At each iteration all edges of the spanning forest are removed. From the above discussion it follows that each node that still has neighbors in the beginning of an iteration, has at least one edge removed during that iteration, and therefore its degree decreases. Hence, the first phase of the algorithm terminates in at most Δ iterations.

The second phase terminates in at most Δ iterations as well. Each iteration consists of two stages. First, the current forest is colored using procedure *3-Color-Rooted-Tree*, which takes, by theorem 2, $O(\lg \Delta \lg^* n)$ time on an EREW PRAM (the $\lg \Delta$ factor appears because we do not use the concurrent-read capability). Now we iterate over all the colors. Since in this section we assume that Δ is a constant, each iteration can be done in $O(\lg \Delta)$ time using word operations. Hence, one iteration of the second phase takes $O(\lg \Delta \lg^* n + \Delta \lg \Delta)$ time, leading to an overall $O(\Delta \lg \Delta(\Delta + \lg^* n))$ running time on an EREW PRAM. ∎

Having a $(\Delta+1)$-coloring of a graph enables us to find an MIS in this graph. The following theorem states this fact formally. (We refer to the algorithm described in the proof as *Constant-Degree-MIS* in the subsequent sections.)

Theorem 4 *An MIS in constant-degree graphs can be found in $O(\lg^* n)$ time on an EREW PRAM using $O(n)$ processors.*

Proof: After coloring the graph in a constant number of colors using the procedure *Color-Constant-Degree-Graph*, one can find an MIS by iterating over the colors, taking all the remaining nodes of the current color, adding them to the independent set, and removing them and all their neighbors from the graph. By theorem 3, the coloring of a constant-degree graph takes $O(\lg^* n)$ time on an EREW PRAM. The selection of all nodes with a specific color and the removal of all neighbors of the selected nodes takes constant time. ∎

The proofs of theorems 3 and 4 also imply that the algorithms *Color-Constant-Degree-Graph* and *Constant-Degree-MIS* have a polylogarithmic running times for graphs with polylogarithmic maximum degrees. However, in this case the assumption that the word size is greater then Δ is unreasonable, so the running time of the algorithms becomes $O(\Delta(\Delta^2 + \lg \Delta \lg^* n))$. In section 6 we present an algorithm with better performance for $\Delta = \omega(\lg n)$.

The above algorithms can be implemented in the distributed model of computation [1,7], where processors have fixed connections determined by the input graph. The algorithms in the distributed model achieve the same $O(\lg^* n)$ bound as in the EREW PRAM model. Linial has recently shown [14] that $\Omega(\lg^* n)$ time is required in the distributed model to find a maximal independent set on a chain. Our algorithms are therefore optimal (to within a constant factor) in the distributed model.

5 Algorithms for Planar Graphs

Any planar graph can be 4-colored. However, linear time sequential algorithms are known only for 5-coloring planar graphs. In this section we describe a simple and efficient parallel algorithm that 7-colors a planar graph, and show how to construct a more complicated parallel algorithm to 5-color a planar graph.

```
PROCEDURE 7-Color-Planar-Graph
V' ← V
V_1, V_2, ... V_{lg n} ← ∅
i ← 0
while V' ≠ ∅ for all v ∈ V' do in parallel ;;; first stage
      if Degree(v) ≤ 6
        then do
               V_i ← V_i + v
               V' ← V' − v
        end
      i ← i + 1
end
for  i ← i − 1 to 0 do ;;; second stage
     while V_i ≠ ∅ do
           E_i ← {(v, w) | v, w ∈ V_i ; (v, w) ∈ E}
           I ← Constant-Degree-MIS(V_i, E_i)
           for  all v ∈ I do in parallel
                C_v ← max{{1 ... 7} − {C_w | w ∈ V'; (v, w) ∈ E} }
                end
           V' ← V' + I
           V_i ← V_i − I
     end
end
```

Figure 5: The 7-Coloring Algorithm For Planar Graphs

First we describe an algorithm for 7-coloring of planar graphs. The algorithm, called *7-Color-Planar-Graph*, is shown in Figure 5. The algorithm consists of two stages. In the first stage, we iteratively partition the vertices of the graph into layers. At each iteration we create a new layer consisting of all nodes of the graph with degree 6 or less and delete these nodes from the graph.

The second stage returns the layers to the graph in the order opposite to the order in which the layers are removed. After a layer is returned, it is 7-colored in the way consistent with the coloring of the layers which have been returned and colored in the previous iterations. Note that all the nodes of the returned layer have a degree of at most 6 in the current graph.

The layer is colored by iteratively applying the *Constant-Degree-MIS* procedure to find an MIS in the subgraph induced by the uncolored nodes of the layer, and coloring each of the selected nodes in a color different from its colored neighbors. Since the uncolored nodes have a degree of at most 6 in the current graph, we never need more than 7 colors.

Theorem 5 *The algorithm* 7-Color-Planar-Graph *runs in* $O(\lg n \lg^* n)$ *time on a CRCW PRAM and in* $O(\lg^2 n)$ *time on an EREW PRAM.*

Proof: In a planar graph, at least a constant fraction ($1/7^{\text{th}}$) of nodes have a degree less or equal to 6, and therefore the first stage of the *7-Color-Planar-Graph* algorithm terminates in at most $O(\lg n)$ steps. At each step we have to identify the nodes that have degree less than 7 in the remaining graph. This takes constant time on a CRCW PRAM (assuming that if two or more processors simultaneously write into some location, one of them will succeed) and $O(\lg n)$ time on an EREW PRAM.

In the second stage all the uncolored nodes are of degree less or equal to 6 and therefore, by theorem 4, the procedure *Constant-Degree-MIS* finds, in $O(\lg^* n)$ time, an MIS in the graph induced by these nodes. By the definition of the maximal independent set, when the algorithm colors the MIS, at least one uncolored neighbor of each uncolored node becomes colored. Therefore the second part of the second

stage terminates in at most 7 iterations.

Since the first stage takes $O(\lg n)$ time on a CRCW PRAM and $O(\lg^2 n)$ time on an EREW PRAM, and since each one of the $O(\lg n)$ iterations of the second stage is dominated by a call to *Constant-Degree-MIS*, the total running time is $O(\lg n \lg^* n)$ on a CRCW PRAM and $O(\lg^2 n)$ on an EREW PRAM. ∎

Remark: If, at each stage, instead of removing from the graph all the nodes with degree less than 6, we remove all the nodes with degree less or equal to the average degree, the algorithm described above produces a correct result in polylogarithmic time for any graph G such that the average degree of any node-induced subgraph G' of G is polylogarithmic in the size of G'. This class contains many important subclasses including graphs that are unions of a polylogarithmic number of planar graphs (i.e. graphs with polylogarithmic thickness).

Our techniques together with the ideas presented in [4] can be used to construct a deterministic $O(\log^3 n \lg^* n)$ time algorithm for 5-coloring a planar graph.

The 5-coloring algorithm has two stages. The first stage of the algorithm partitions the graph into layers such that vertices in any layer are independent and have degree of at most 6 in the graph induced by the vertices in its layer and the higher numbered layers. The second stage of the algorithm adds layers one by one, starting from the layer with the highest number, each time recoloring the graph with 5 colors.

Before describing the second stage, we need the following definitions. Let G be a partially colored graph and let c_1 and c_2 be two distinct colors. A *color component* is a connected component of a subgraph of G induced by all vertices of color c_1 and c_2. A *color component flip* is a recoloring of the color component that exchanges colors c_1 and c_2. A color component flip does not affect the validity of coloring.

We can proceed with the description of the second stage of the algorithm. After a layer is added to already colored graph, we first color all vertices that can be colored without changing the existing coloring. This can be done in the same way as in the 7-coloring algorithm. Now all 5 colors are represented among neighbors of each

uncolored vertex. Since the uncolored vertices have degree of at most 6, the results of [4] imply that for every uncolored vertex v there are two colors c_1 and c_2 such that v has exactly one neighbor w_1 of color c_1 and exactly one neighbor w_2 of color c_2. Furthermore, the vertices w_1 and w_2 belong to different color components induced by colors c_1 and c_2. Flipping each one of these color component allows us to color v. The problem is, however, that flipping both color components simultaneously does not allow us to color v. We call such color components *dependent*.

Where as Boyar and Karloff use randomness to deal with this problem, we use our symmetry-breaking techniques as follows. For each pair of distinct colors c_1 and c_2, we construct color components induced by these colors. Then we construct a *dependency graph* with vertices corresponding to the color components and edges corresponding to the dependencies between the color components. Flipping a set of color components that corresponds to an independent set in the dependency graph does not cause conflicts. Suppose we can find an independent set in the dependency graph such that flipping the corresponding set of color components allows us to color a constant set of uncolored vertices. Then in $O(\log n)$ iterations will be able to color all uncolored vertices.

We find such an independent set in the dependency graph as follows. Observe that the dependency graph is planar, so we can 7-color this graph using the 7-Color-Planar-Graph algorithm. Then, for each pair of distinct colors and for each color class of the corresponding dependency graph, we compute the number of uncolored vertices of the original graph which can be colored if the color components corresponding to vertices in the color class are flipped. For each of the 10 possible choices of colors c_1 and c_2 there are 7 color classes, so the total number of times that we count the number of vertices that can be colored if a color class is flipped is 70. Since each uncolored vertex is counted at least once, there is a color class such that flipping all color components in this class allows us to color at least $1/70$ uncolored vertices.

Next we analyze to complexity of the algorithm. The outer loop of the algorithm that iterates over layers is executed $O(\log n)$ times, and the inner loop that colors a constant fraction of uncolored vertices is executed $O(\log n)$ times as well.

Each iteration of the inner loop does 10 connected component computations, 70 enumeration and 10 calls to the 7-Color-Planar-Graph procedure. Since each connected component computation can be done in $O(\log n)$ time on CRCW PRAM using Shiloach-Vishkin algorithm [17], the 7-Color-Planar-Graph procedure is the bottleneck of the inner loop (recall that it runs in $O(\log n \lg^* n)$ time). The overall running time of the algorithm is $O(\log^3 n \lg^* n)$.

The above result is summarized in the following theorem.

Theorem 6 *A planar graph can be 5-colored in $O(\lg^3 n \lg^* n)$ time on a CRCW PRAM using $O(n)$ processors.*

Using the techniques described in this paper it is easy to construct a fast algorithm for finding a maximal matching in planar graph.

Theorem 7 *A maximal matching in planar graph can be found in $O(\lg n \lg^* n)$ time on a CRCW PRAM.*

Proof: First, the algorithm partitions the graph into layers, such that the nodes in a layer are of degree less than 7 in the graph induced by the nodes of this layer and the nodes in the higher-numbered layers. The algorithm proceeds by iteratively returning a layer, finding a maximal matching in the obtained graph, and removing the end-points of the edges in the matching. At the end of each iteration the remaining nodes induce a graph of degree zero and therefore at the beginning of each iteration the maximum degree of the induced graph is 6. Hence, a maximal matching in this graph can be found in $O(\lg^* n)$ time by finding a maximal independent set in the line-graph, which also has a constant maximum degree. Each iteration takes $O(\lg^* n)$ time on a CRCW PRAM and the number of iterations is $O(\lg n)$. This gives $O(\lg n \lg^* n)$ total running time. ∎

6 Coloring Polylogarithmic Degree Graphs

This section describes a coloring algorithm for graphs with maximum degree which is polylogarithmic in the size of the graph. For $\Delta = \omega(\lg n)$, this algorithm has a better performance than the algorithm *Color-Constant-Degree-Graph* described above.

The *Poly-Log-Color* algorithm is shown in Figure 6 and works as follows. First, the graph is partitioned into two subgraphs with approximately equal number of nodes, and the subgraphs are recursively colored in $\Delta+1$ colors. Then we iterate through all the colors of one of the subgraphs, recoloring each node with a color different from the colors of all of its neighbors.

Theorem 8 *The algorithm* Poly-Log-Color *colors a graph with a maximum degree of* Δ *with* $\Delta+1$ *colors in* $O(\Delta^2 \lg n)$ *time.*

Proof: Each time the graph is partitioned into two subgraphs with approximately equal number of nodes and therefore the depth of recursion is $O(\lg n)$. At each recursion level we iterate through all the colors, each iteration dominated by the time to find a color different from the colors of all the neighbors of a node, which takes $O(\Delta)$ time. Hence the total time is $O(\Delta^2 \lg n)$ on a EREW PRAM. ∎

After coloring the graph in $\Delta+1$ colors we can construct an MIS of the graph in $O(\Delta^2)$ time. Hence, an MIS of a graph with a polylogarithmic maximum degree can be found in $O(\Delta^2 \lg n)$ time on EREW PRAM using a linear number of processors.

7 Lower Bounds

In this section we prove two lower bounds for a CRCW PRAM with polynomial number of processors:

- Finding a MIS in a general graph takes $\Omega(\lg n / \lg\lg n)$ time.

```
PROCEDURE Poly-Log-Color (V, E)
partition V into V_r,V_l such that V_r ∪ V_l = V
E_r ← {(v, w) | (v, w) ∈ E; v, w ∈ V_r}
E_l ← {(v, w) | (v, w) ∈ E; v, w ∈ V_l}
C_r ←Poly-Log-Color(V_r, E_r)
C_l ←Poly-Log-Color(V_l, E_l)
V' ← ∅
for all v ∈ V_l in parallel do
    if ∃(v, w) ∈ E such that v ∈ V_l, w ∈ V_r and C_l(v) = C_r(w)
      then do
            V' ← V' ∪ v
      end
for j ← 1 to Δ+1 do
    for all v ∈ V' in parallel do
        if C_l(v) = j
          then do
                C_l(v) ← max {{1, 2, ...Δ+1} − {C(w) | (v, w) ∈ E'}}
                V' ← V' − v
          end
    end
end
```

Figure 6: The Coloring Algorithm for Polylogarithmic Maximum Degree Graphs

- 2-coloring a directed list takes $\Omega(\lg n / \lg\lg n)$ time.

The first lower bound complements the $O(\lg n)$ CRCW PRAM upper bound for the MIS problem that is achieved by Luby's algorithm [15]. The second lower bound complements Theorem 2 in this paper.

Theorem 9 *The running time of any MIS algorithm on a CRCW PRAM with a polynomial number of processors is* $\Omega(\lg n / \lg\lg n)$.

Proof: Given an instance of MAJORITY, we construct an instance of MIS in constant CRCW PRAM time. MAJORITY is harder that PARITY [6], which was proven to take $\Omega(\lg n / \lg\lg n)$ on a CRCW PRAM in [2,3]. Therefore the lower bound claimed in the theorem follows.

Let $x_1, x_2, \ldots, x_n$ be an instance of MAJORITY. We construct a complete bipartite graph $G = (V, E)$ with nodes corresponding to '0' bits of the input on one side and nodes corresponding to '1' bits on the other side.

$$\begin{aligned} V &= \{1, \ldots, n\} \\ E &= \{(i,j) \mid x_i \neq x_j\} \end{aligned}$$

To construct this graph, assign a processor P_{ij} for each pair $1 \leq i < j \leq n$. Then, each processor P_{ij} writes 1 into location M_{ij} if $x_i \neq x_j$ and 0 otherwise.

A maximal matching in a complete bipartite graph is also a maximum one. By constructing a maximal independent set in the line-graph G' of G, one can find a maximal matching in G. To construct the graph G' assign a processor P_{ijk} for each distinct $i, j, k \leq n$. Each P_{ijk} writes 1 into location $M_{(i,j),(j,k)}$ if $M_{ij} = M_{jk} = 1$ and 0 otherwise.

The MAJORITY equals to 1 if and only if there is an unmatched node $i \in G$ such that $x_i = 1$, which can be checked on a CRCW PRAM in constant time. ∎

Theorem 10 *The time to 2-color a directed list on a CRCW PRAM with a polynomial number of processors is* $\Omega(\lg n / \lg\lg n)$.

Proof: We show a constant time reduction from PARITY to the 2-coloring of a directed list. First, we show how to construct, in constant time, a directed list with elements corresponding to all the input bits x_i with value of 1. Let $x_1, x_2, \ldots, x_n$ be an instance of PARITY. Associate a processor P_i with each input cell M_i that initially holds the value of x_i. Associate a set of processors P_i^{jk} with each index i, $1 \leq k \leq j < i$. In one step, each processor P_i^{jk} reads the value of M_k and, if it equals to 1, writes 1 into M_i^j, effectively computing the OR-function on the input values $x_{i-j}, x_{i-j+1}, \ldots, x_{i-1}$. Assign a processor P_i^j to each M_i^j. Each processor P_i^j reads M_i^j and M_i^{j+1} and writes j into M_i' if and only if $M_i^j \neq M_i^{j+1}$. It can be seen that for all $0 \leq i \leq n$, M_i' holds $\max\{j \mid j < i, x_j = 1\}$.

We have constructed a directed list with elements corresponding to all the input bits x_i with value of 1. Assume this list is 2-colored. Then PARITY equals to 1 if and only if both ends of the list are colored in the same color, which can be checked in constant time. ∎

8 Acknowledgments

We would like to thank Charles Leiserson and David Shmoys for fruitful and stimulating discussions, and for their valuable comments on a draft of this paper.

References

[1] B. Awerbuch. Complexity of network syncronization. *Journal of the Association for Computing Machinery*, 32(4):804–823, October 1985.

[2] P. Beame. *Lower Bounds in Parallel Machine Computation*. PhD thesis, University of Toronto, 1986.

[3] P. Beame and J. Hastad. Personal communication. 1986.

[4] J. Boyar and H. Karloff. Coloring planar graphs in parallel. 1986. Unpublished Manuscript.

[5] R. Cole and U. Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In *Proc. 18th ACM Simp. on Theory of Computing*, pages 206–219, 1986.

[6] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. In *Proc. 22nd IEEE Conf on Foundations of Computer Science*, pages 260–270, 1981.

[7] R. G. Gallager, P. A. Humblet, and P. M Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

[8] A. Goldberg and S. Plotkin. Parallel $(\Delta+1)$ coloring of constant-degree graphs. *Information Processing Letters*, 1986. Accepted for publication.

[9] M. Goldberg and T. Spencer. A new parallel algorithm for the maximal independent set problem. 1986. Submitted for publication.

[10] A. Israeli and Y. Shiloach. An improved parallel algorithm for maximal matching. *Information Processing Letters*, 22:57–60, January 1986.

[11] H. J. Karloff. *Fast Parallel Algorithms for Graph-Theoretic Problems: Matching, Coloring, Partitioning.* PhD thesis, University of California, Berkeley, 1985.

[12] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. In *Proc. 16th ACM Simp. on Theory of Computing*, pages 266–272, 1984.

[13] C. Leiserson and B. Maggs. Communication-efficient parallel graph algorithms. In *Proc. of International Conference on Parallel Processing*, pages 861–868, 1986.

[14] N. Linial. Personal communication. 1986.

[15] M. Luby. A simple parallel algorithm for the maximal independent set problem. In *Proc. 17th ACM Simp. on Theory of Computing*, pages 1–10, 1985.

[16] G. Shannon. Reduction techniques for designing linear-processor parallel algorithms on sparse graphs. 1986. In prepapation.

[17] Y. Shiloach and U. Vishkin. An $O(\log n)$ parallel connectivity algorithm. *Journal of Algorithms*, 3:57–67, 1982.

[18] L. G. Valiant. Parallel computation. In *Proc. 7th IBM Simp. on Mathemetical Foundations of Computer Science*, 1982.

Efficient Multichip Partial Concentrator Switches

Thomas H. Cormen

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

February, 1987

Abstract

Due to chip area and pin count constraints, large concentrator switches sometimes must be partitioned among several chips. This paper presents designs for two multichip partial concentrator switches, both of which follow from a lemma showing that an ε-nearsorter is also an $(n, m, 1 - \varepsilon/m)$ partial concentrator.

The first switch, based on the Revsort algorithm, is an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch with at most $2\sqrt{n} + \lceil (\lg n)/2 \rceil$ data pins per chip, $\Theta(\sqrt{n})$ chips, and volume $\Theta(n^{3/2})$. A message incurs $3 \lg n + O(1)$ gate delays in passing through the switch.

The second switch, based on Columnsort, is an $(n, m, 1 - O(n^{2-2\beta}/m))$ partial concentrator switch with $\Theta(n^{\beta})$ data pins per chip, $\Theta(n^{1-\beta})$ chips, and volume $\Theta(n^{1+\beta})$, for any $1/2 \leq \beta \leq 1$. A message incurs $4\beta \lg n + O(1)$ gate delays.

1 Introduction

The problem of concentrating relatively few signals on many input lines onto a lesser number of output lines must be solved in many kinds of communication networks. In many parallel computing systems, information is packaged into messages which are routed among the processors. The switches that route these messages sometimes require more chip area or input and output wires than a single chip can supply. This paper presents two designs for fast multichip partial concentrator switches suitable for routing bit-serial messages in a parallel supercomputer. The key lemma of this paper may be used to justify other partial concentrator designs.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-80-C-0622 and in part by a National Science Foundation Fellowship.

An *n-by-m perfect concentrator switch* has n input wires $X_1, X_2, \ldots, X_n$ and $m \leq n$ output wires $Y_1, Y_2, \ldots, Y_m$. The switch can establish m disjoint electrical paths from any set of m input wires to the m output wires. A perfect concentrator switch always routes as many messages as possible. Specifically, whenever k out of the n input wires of an n-by-m perfect concentrator switch carry messages, one of the following is true:

- If $k \leq m$, then an electrical path is established from each input wire that contains a message to an output wire.
- If $k > m$, then each output wire has an electrical path established from an input wire that contains a message.

When $k > m$, some messages cannot be successfully routed, in which case we say the switch is *congested*. Typical ways of handling unsuccessfully routed messages in a routing network are to buffer them, to misroute them, or to simply drop them and rely on a higher-level acknowledgment protocol to detect this situation and resend them. The switch designs in this paper are compatible with any of these congestion control methods.

One way to create a perfect concentrator switch is with a hyperconcentrator switch. An *n-by-n hyperconcentrator switch* has n input wires $X_1, X_2, \ldots, X_n$ and n output wires $Y_1, Y_2, \ldots, Y_n$. The switch can establish disjoint electrical paths from any set of k input wires, for any $1 \leq k \leq n$, to the first k output wires $Y_1, Y_2, \ldots, Y_k$. In other words, we route the k messages to the first k output wires. We can make any n-by-m perfect concentrator switch from an n-by-n hyperconcentrator switch by simply choosing the first m output wires of the hyperconcentrator switch, $Y_1, Y_2, \ldots, Y_m$, as the m output wires of the perfect concentrator switch.

An efficient n-by-n hyperconcentrator switch design is given in [1] and [2]. This switch has a highly regular layout in both ratioed nMOS and domino CMOS technologies, and a signal incurs exactly $2 \lg n$ gate delays through the switch.[1] This switch uses $\Theta(n^2)$ components and has area $\Theta(n^2)$.

Partitioning this hyperconcentrator switch among multiple chips with p pins each requires $\Omega((n/p)^2)$ chips, since each p-pin chip has area $O(p^2)$ and there are $\Theta(n^2)$ components to partition. We may need to partition the switch for two reasons:

1. The $\Theta(n^2)$ area may exceed the available chip area.

2. If the switch is to be packaged by itself on a chip, it may require more input and output pins than are provided by the packaging technology.

A different hyperconcentrator switch, comprised of a parallel prefix circuit and a butterfly network [1], can be built in volume $\Theta(n^{3/2})$ with $O(n \lg n)$ chips and as few as four data pins per chip, but this switch is not combinational. Although its sequential control is not very complex, it is not as simple as that of a combinational circuit.

Partial concentrator switches, as we shall see in Sections 4 and 5, can be combinational with relatively low gate delays. Yet, given chips with p pins, we can partition n-input partial concentrator switches using only $\Theta(n/p)$ chips. An (n, m, α) *partial concentrator switch* has n input wires $X_1, X_2, \ldots, X_n$, $m \leq n$ output wires $Y_1, Y_2, \ldots, Y_m$, and a fraction $0 < \alpha \leq 1$ such that disjoint electrical paths may be established from any set of k input wires, for any $1 \leq k \leq \alpha m$, to k output wires.

A lightly loaded partial concentrator switch is similar to a perfect concentrator switch. If there are k messages entering an (n, m, α) partial concentrator switch, one of the following is true:

- If $k \leq \alpha m$, then an electrical path is established from each input wire that contains a message to an output wire.

- If $k > \alpha m$, then at least αm electrical paths are established from input wires containing messages to output wires.

We call the fraction α the *load ratio*. If a partial concentrator switch is lightly loaded, i.e., the number of messages entering is at most αm, then *all* the messages are routed to output wires.

[1] We use the notation $\lg n$ to denote $\log_2 n$.

An $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch can be used anywhere an n-by-m perfect concentrator switch is required. Consider a set of $k \leq m$ messages to be routed through an n-by-m perfect concentrator switch. For the $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch, we have that $k \leq m = \alpha \cdot (m/\alpha)$, and thus all k messages are routed to output wires. If there are instead $k > m$ messages to be routed through the perfect concentrator switch, we have that $k > m = \alpha \cdot (m/\alpha)$ for the $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch, and thus m output wires carry messages. In either case, the partial concentrator switch performs the same function as the perfect concentrator switch, at the cost of a $1/\alpha$-factor increase in the number of input and output wires.

In this paper, we show a connection between nearsorting and partial concentration. We then use this relationship to design two efficient multichip partial concentrator switches, both of which use the hyperconcentrator switch of [1] and [2] as a subcircuit on a single chip.

The remainder of this paper is organized as follows. Section 2 covers some basic terminology and describes the message format upon which the switches are based. Section 3 defines nearsorting and shows the relationship between nearsorting and partial concentration. Section 4 presents a design for a partial concentrator switch based on the Revsort algorithm for sorting on a mesh; Section 5 does the same, but based on the Columnsort algorithm for sorting on a mesh. Finally, Section 6 contains further remarks about multichip concentrator switches.

2 Preliminaries

In this section, we define some basic terminology and mathematical conventions and present the message format assumed by the switch designs.

Bit and boolean values are denoted by "1" and "0" for TRUE and FALSE respectively.

We assume that the switches route *bit-serial messages*. Each message is formed by a stream of bits arriving at a wire at the rate of one bit per clock cycle. The first bit of each message that arrives at an input wire is the *valid bit*, indicating whether subsequent bits arriving on that wire form a valid message or an invalid message. The bit sequence following a valid bit of 1 forms a *valid message*, which we would like to be routed from an input wire to an output wire of the switch. From there it may pass through the remainder of the routing network. A valid bit of 0 indicates an *invalid message*, which does not need to be routed to an output wire.

The valid bits all arrive at the input wires of a switch during the same clock cycle, which we call *setup*. An external control line signals setup. Message bits entering through input wires at cycles after setup follow the electrical paths in the switch that are established during setup.

We shall adopt some notational conventions to ease the exposition in the remainder of this paper. Uppercase symbols denote wire names and lowercase symbols denote integer values. We shall also use uppercase symbols to denote bit values on the wires they name when the usage is unambiguous. Wire names will usually be subscripted.

A sequence of values is *sorted* if it is in nonincreasing order. The valid bits output by an n-by-n hyperconcentrator switch are thus sorted, since if there are k valid messages, we have

$$\begin{aligned} Y_1, Y_2, \ldots, Y_k &= 1 \\ Y_{k+1}, Y_{k+2}, \ldots, Y_n &= 0 \end{aligned}$$

during setup.

Concentrators were originally presented as graphs in, for example, [4,5,8]. The term "hyperconcentrator" is due to Valiant. Vertex-disjoint paths from designated input nodes to designated output nodes are the concentrator graph counterpart of the combinational routing paths established during setup in the concentrator switches of this paper.

3 Nearsorting and Partial Concentration

In this section, we define ε-nearsorting and show its relationship to partial concentration. The key lemma proven in this section is used in the next two sections to justify partial concentrator switch constructions.

A sequence of values is *ε-nearsorted* if each element in the sequence is within ε positions of where it belongs in the fully sorted sequence. For example, the sequence $5, 3, 6, 1, 4, 2$ is 2-nearsorted since each element is at most two places away from its correct position in the fully sorted sequence $6, 5, 4, 3, 2, 1$. The value ε need not be a constant; we will usually let ε be a function of the size of the sequence. A fully sorted sequence is also 0-nearsorted.

Since we are only interested in nearsorting valid bits, for the remainder of this paper we shall be concerned only with inputs whose value is either 0 or 1. We say that a sequence of values is *clean* if they all have the same value; otherwise the sequence is *dirty*. The following lemma describes an ε-nearsorted sequence of 0's and 1's.

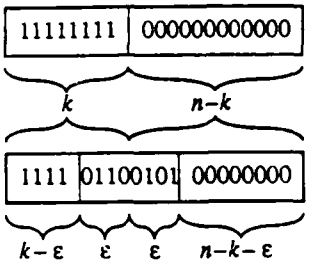


Figure 1: A fully sorted sequence of k 1's and $n-k$ 0's and an ε-nearsorted sequence of the same values. The ε-nearsorted sequence consists of a clean sequence of at least $k-\varepsilon$ 1's followed by a dirty sequence of at most 2ε bits followed by a clean sequence of at least $n-k-\varepsilon$ 0's.

Lemma 1 *A sequence of n bits, containing k 1's and $n-k$ 0's, is ε-nearsorted if and only if it consists of a clean sequence of at least $k-\varepsilon$ 1's followed by a dirty sequence of at most 2ε bits followed by a clean sequence of at least $n-k-\varepsilon$ 0's.*

Proof ($\Rightarrow$) As shown in Figure 1, a fully sorted sequence of k 1's and $n-k$ 0's is simply k 1's followed by $n-k$ 0's. In an ε-nearsorted sequence of the same values, each 1 appears within the first $k+\varepsilon$ positions, and each 0 appears within the last $n-k+\varepsilon$ positions. The only dirty sequence within the ε-nearsorted sequence is therefore centered at the kth position and extends ε positions to either side. The lemma then follows.

($\Leftarrow$) Again referring to Figure 1, each 1 is within the first $k+\varepsilon$ positions, and each 0 is within the last $n-k+\varepsilon$ positions. The sequence is thus ε-nearsorted. □

The following lemma is the key lemma that relates ε-nearsorting to partial concentration.

Lemma 2 *Let P be a switch with n inputs $X_1, X_2, \ldots, X_n$ and n outputs $Y_1, Y_2, \ldots, Y_n$, and suppose that P ε-nearsorts valid bits. Then by restricting the outputs of P to $Y_1, Y_2, \ldots, Y_m$, for any $m \leq n$, P is an (n, m, α) partial concentrator switch, where $\alpha = 1 - \varepsilon/m$.*

Proof Consider any input to switch P containing k 1's and $n-k$ 0's. We have $\alpha m = (1-\varepsilon/m)m = m-\varepsilon$, and there are two cases.

Case 1: $k \leq \alpha m = m - \varepsilon$. We have $m \geq k + \varepsilon$. Since P is an ε-nearsorter, each 1 appears within the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\} \subseteq \{Y_1, Y_2, \ldots, Y_m\}$. Thus, each 1 is routed to an output of the partial concentrator switch.

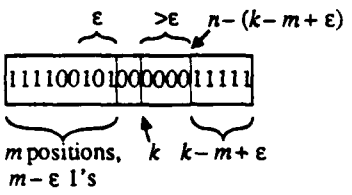


Figure 2: The output of an $(n, m, 1-\varepsilon/m)$ partial concentrator switch that is not ε-nearsorted. This switch routes $m-\varepsilon$ out of $k > m-\varepsilon$ 1's to the first m outputs, but the remaining $k-m+\varepsilon$ 1's are routed to the last $k-m+\varepsilon$ out of the n outputs. If we have $k+\varepsilon < n-(k-m+\varepsilon)$, or equivalently, $k+\varepsilon < (n+m)/2$, then the last $k-m+\varepsilon$ 1's are not within ε positions of output Y_k, and thus the output sequence is not ε-nearsorted.

Case 2: $k > \alpha m = m-\varepsilon$. We have $m < k+\varepsilon$. Again, each 1 appears within the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\}$. From Lemma 1, we know that at most ε of the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\}$ carry 0's, so at most ε of the outputs $\{Y_1, Y_2, \ldots, Y_m\}$ carry 0's. Thus, at least $m-\varepsilon = \alpha m$ of the outputs $\{Y_1, Y_2, \ldots, Y_m\}$ carry 1's.

We conclude that by restricting the outputs of P to $Y_1, Y_2, \ldots, Y_m$, P is an $(n, m, 1-\varepsilon/m)$ partial concentrator switch. □

The converse of Lemma 2 is not necessarily true. As shown in Figure 2, if an $(n, m, 1-\varepsilon/m)$ partial concentrator switch routes $m-\varepsilon$ out of $k > \alpha m = m-\varepsilon$ 1's to the first m outputs, the remaining $k-m+\varepsilon$ 1's may be routed to the last $k-m+\varepsilon$ out of the n outputs. In this case, if there are more than ε outputs between Y_k and $Y_{n-(k-m+\varepsilon)}$, then the output sequence is not ε-nearsorted.

4 A Revsort-Based Partial Concentrator Switch

In this section, we present a design for an (n, m, α) partial concentrator switch that uses $\Theta(\sqrt{n})$ chips with only $\Theta(\sqrt{n})$ data pins each. The basic building block is the hyperconcentrator switch of [1] and [2] placed on a chip. Each message incurs $3\lg n + O(1)$ gate delays in passing through the switch. The load ratio is $\alpha = 1-O(n^{3/4}/m)$. Most of the results of this section originally appeared in [1].

This partial concentrator switch can be implemented in

- two dimensions with $\Theta(n^2)$ area and one chip type with $2\sqrt{n}$ data pins, or
- three dimensions with $\Theta(n^{3/2})$ volume, two chip types with at most $2\sqrt{n}+\lceil(\lg n)/2\rceil$ pins, and two board types.

The design is based on Schnorr and Shamir's Revsort algorithm for sorting on a mesh [7], which, although not optimal for sorting on a mesh, is simple. The idea behind the partial concentrator switch is to nearsort a $\sqrt{n}$-by-$\sqrt{n}$ matrix of valid bits. The m output wires of the switch correspond to the first m nearsorted matrix entries.

We need some basic definitions. We assume that the rows and columns of the $\sqrt{n} \times \sqrt{n}$ matrix are numbered $0, 1, \ldots, \sqrt{n}-1$ and that $\sqrt{n} = 2^q$ for some integer q. We also define, for any integer i, $0 \le i < \sqrt{n}$, $rev(i)$ to be the binary number obtained by reversing the q bits in the binary representation of i, including the leading zeros. For example, when $\sqrt{n} = 16$, $rev(3)$ is 12.

The partial concentrator switch is built from three stages, each stage containing $\sqrt{n}$ hyperconcentrator chips. Each $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip serves to fully sort a row or column of valid bits in the underlying matrix. We shall denote by $H_{l,i}$ the ith hyperconcentrator chip in stage l, for $1 \le l \le 3$ and $0 \le i < \sqrt{n}$, with input wires $X_{l,i,0}, X_{l,i,1}, \ldots, X_{l,i,\sqrt{n}-1}$ and output wires $Y_{l,i,0}, Y_{l,i,1}, \ldots, Y_{l,i,\sqrt{n}-1}$.

The general idea of the construction of the partial concentrator switch is as follows. Each stage 1 chip corresponds to a column of the matrix, so the stage 1 chips fully sort the valid bits in each column. The input and output wires $X_{1,j,i}$ and $Y_{1,j,i}$ represent the value of the matrix element at row i and column j before and after sorting.

The wiring between stages 1 and 2 is effectively a matrix transposition, accomplished by connecting the output wire $Y_{1,j,i}$ to the input wire $X_{2,i,j}$ for $0 \le i, j < \sqrt{n}$. Each stage 2 chip then corresponds to a row of the matrix, so the stage 2 chips fully sort the valid bits in each row. The input and output wires $X_{2,i,j}$ and $Y_{2,i,j}$ represent the value of the matrix element at row i and column j before and after sorting.

The wiring between stages 2 and 3 is the composition of two matrix permutations. We first cyclically rotate row i by $rev(i)$ places to the right, for $0 \le i < \sqrt{n}$. That is, the matrix element in row i and column j, for $0 \le i, j < \sqrt{n}$, is moved to row i and column $(rev(i)+j) \bmod \sqrt{n}$. The matrix is then transposed. Each stage 3 chip then corresponds to a column of the matrix, so the stage 3 chips fully sort the valid bits in each column. The two permutations are accomplished in one wiring step by connecting the output wire $Y_{2,i,j}$ to the input wire $X_{3,(rev(i)+j) \bmod \sqrt{n},i}$, for $0 \le i, j < \sqrt{n}$.

The output wires of the partial concentrator switch are the first m output wires of the matrix in row-major order, or $Y_{3,j,i}$ for $0 \le i < \lfloor m/\sqrt{n} \rfloor$ and $0 \le j < \sqrt{n}$ or $i = \lfloor m/\sqrt{n} \rfloor$ and $0 \le j < m \bmod \sqrt{n}$.

Like the hyperconcentrator chips from which it is built, the partial concentrator switch is a combinational circuit. The routing paths are established by the valid bits during setup, and subsequent bits follow along these paths.

To see that this construction does indeed yield an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch, we first observe that its operation is equivalent to the following algorithm, which corresponds to the first $1\frac{1}{2}$ iterations of Revsort:

Algorithm 1 Given a $\sqrt{n} \times \sqrt{n}$ matrix with $\sqrt{n} = 2^q$ and matrix element values of 0 or 1, perform the following four steps:

1. Fully sort the columns.
2. Fully sort the rows.
3. For $0 \le i < \sqrt{n}$, cyclically rotate row i by $rev(i)$ places to the right, i.e., move the element in column j to column $(rev(i) + j) \bmod \sqrt{n}$.
4. Fully sort the columns.

The three sorting steps correspond to the three stages of hyperconcentrator chips in the partial concentrator switch construction. The wiring between stages 1 and 2 corresponds to changing from sorting columns to sorting rows. The wiring between stages 2 and 3 corresponds to the cyclic rotations within rows and changing from sorting rows to sorting columns. We are now ready to prove that this construction works.

Theorem 3 *The Revsort-based construction yields an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch.*

Proof Both [1] and [7] show that after running Algorithm 1 on a $\sqrt{n} \times \sqrt{n}$ matrix with elements valued 0 or 1, the matrix consists of only clean rows of 1's at the top, clean rows of 0's at the bottom, and at most $2\lceil n^{1/4} \rceil - 1$ dirty rows in the middle. Since each row contains $\sqrt{n}$ elements, there are at most $O(n^{3/4})$ dirty bits. By Lemma 1, the sequence is $O(n^{3/4})$-nearsorted, and by Lemma 2, the circuit is an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch. □

Figure 3 shows a two-dimensional layout of the switch using $3\sqrt{n}$ hyperconcentrator chips, with $2\sqrt{n}$ data pins each. We simply use crossbar wiring to permute the wires between hyperconcentrator chips of consecutive stages. The area of this layout is $\Theta(n^2)$ since the crossbar wiring area is $\Theta(n^2)$, which dominates the total chip area of $\Theta(n^{3/2})$. (Each stage of $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chips consists of $\sqrt{n}$ chips, each with area $\Theta(n)$, for a total chip area of $\Theta(n^{3/2})$.)

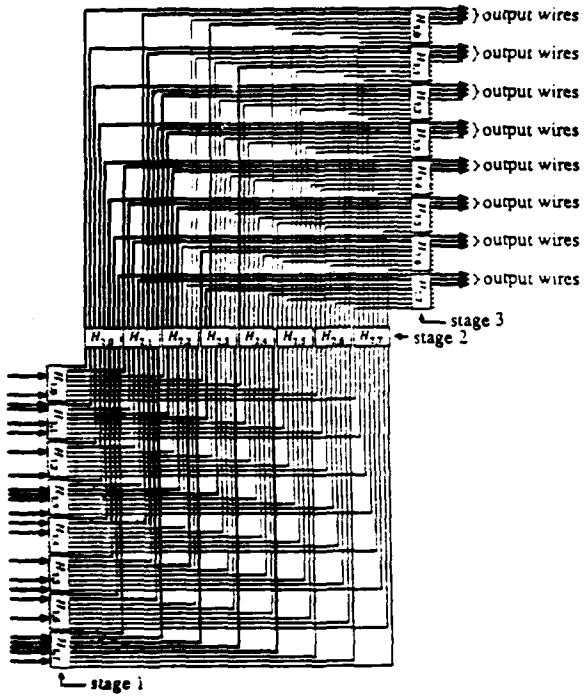


Figure 3: A two-dimensional layout of the Revsort-based partial concentrator switch with $n = 64$ inputs and $m = 28$ outputs. The electrical paths established by 24 valid messages are shown with heavy lines. The output wires are the top four output wires of hyperconcentrator chips $H_{3,0}, H_{3,1}, H_{3,2}, H_{3,3}$ and the top three output wires of hyperconcentrator chips $H_{3,4}, H_{3,5}, H_{3,6}, H_{3,7}$.

A signal incurs $2\lceil \lg \sqrt{n} \rceil + O(1)$ gate delays in passing through each chip. The $2\lceil \lg \sqrt{n} \rceil$ gate delays are from the hyperconcentrator switch within the chip. The I/O pad circuitry accounts for the additional $O(1)$ delay. The total number of gate delays incurred by a signal passing through the entire partial concentrator switch is thus

$$\begin{aligned} 6\lceil \lg \sqrt{n} \rceil + O(1) &\le 6 \lg \sqrt{n} + O(1) \\ &= 3 \lg n + O(1) \,. \end{aligned}$$

As shown in Figure 4, we can package the partial concentrator switch in three dimensions using volume $\Theta(n^{3/2})$. Each circuit board contains one $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip, corresponding to one row or

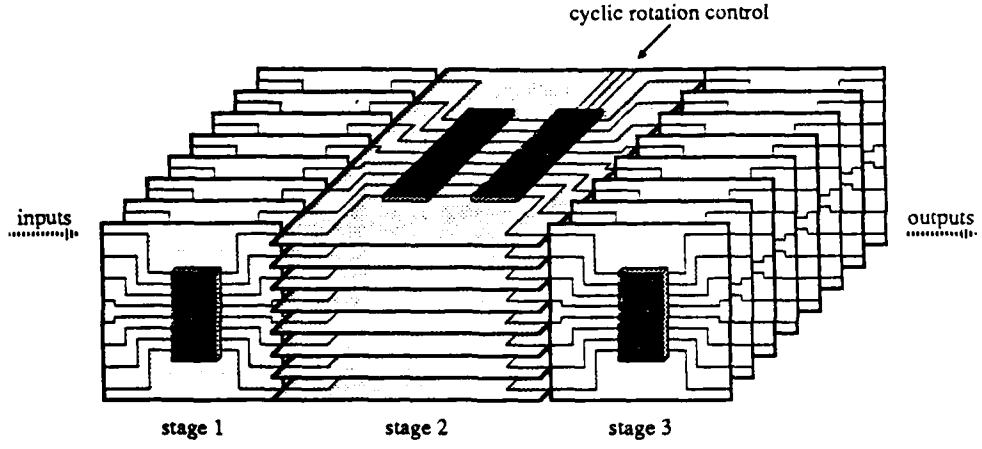


Figure 4: The three-dimensional packaging of the Revsort-based partial concentrator switch for $n = 64$. Each stack contains $\sqrt{n}$ circuit boards and corresponds to one stage. Each board contains one $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip, and boards in stack 2 follow the hyperconcentrator chip by a $\sqrt{n}$-bit barrel shifter chip to perform the cyclic rotation of each row. The $\lg \sqrt{n}$ control bits that determine the shift amount for each barrel shifter are hardwired.

column of the matrix. Each of the three stacks contains $\sqrt{n}$ boards and represents one stage. The wires cross stack junctions in a $\sqrt{n} \times \sqrt{n}$ array, with the valid bit value of the wire in row i and column j equal to the value of the matrix element in the same position at the corresponding step of Algorithm 1.

The matrix transpose between stages 1 and 2 is performed in the natural way, with the ith output wire from board j in stage 1 going straight across the junction to be the jth input wire of board i in stage 2. The wiring permutation between the hyperconcentrator chips of stages 2 and 3 includes the cyclic rotations of the rows, followed by the transpose. The transpose is performed in the natural way once again. We perform the cyclic rotation by following each stage 2 hyperconcentrator chip by a $\sqrt{n}$-bit barrel shifter on the same board. The barrel shifter has $\sqrt{n}$ input wires, $\sqrt{n}$ output wires, and $\lceil \lg \sqrt{n} \rceil$ control bits which, interpreted as a binary integer, determine the rotation amount. We hardwire the control bits in the ith board to have the value $rev(i)$.

We use only two board types, $3\sqrt{n}$ hyperconcentrator chips, and $\sqrt{n}$ barrel shifters in building the switch. All $2\sqrt{n}$ boards in stages 1 and 3 are identical, as are all $\sqrt{n}$ stage 2 boards. The barrel shifters require $2\sqrt{n} + \lceil \lg \sqrt{n} \rceil = 2\sqrt{n} + \lceil (\lg n)/2 \rceil$ data pins. The hardwiring of the barrel shifter control bit values can be performed after the boards have been fabricated.

To see that the volume is $\Theta(n^{3/2})$, we need only consider the stage 2 stack, which has the most components. Each board contains a $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip and a $\sqrt{n}$-bit barrel shifter, both having area $\Theta(n)$. The whole stack of $\sqrt{n}$ boards, and therefore the entire switch, has volume $\Theta(n^{3/2})$.

Since the barrel shift amounts are hardwired and never change, the barrel shifters introduce only a constant number of gate delays. A signal therefore incurs $3 \lg n + O(1)$ gate delays in passing through the three-dimensional switch.

Letting p, the number of pins per chip, be $\Theta(\sqrt{n})$, both the two-dimensional and three-dimensional layouts use only $\Theta(n/p)$ chips.

5 A Columnsort-Based Partial Concentrator Switch

In this section, we present a design for an (n, m, α) partial concentrator switch that uses $\Theta(n^{1-\beta})$ chips with $\Theta(n^{\beta})$ pins each, where $1/2 \leq \beta \leq 1$. The basic building block is a $\Theta(n^{\beta})$-by-$\Theta(n^{\beta})$ hyperconcentrator chip. Each message incurs $4\beta \lg n + O(1)$ gate delays in passing through the switch. The load ratio is $\alpha = 1 - O(n^{2-2\beta}/m)$. This switch can be implemented in two dimensions with area $O(n^2)$ or in three dimensions with volume $\Theta(n^{1+\beta})$. Table 1 shows resource measures for the Revsort-based switch and the values of β at which the switch of this section matches them asymptotically.

The design is based on Leighton's Columnsort algorithm [3] for sorting n elements on an $r \times s$ mesh, where $n = rs$ and s evenly divides r. The idea behind this partial concentrator switch is to $(s-1)^2$-nearsort an $r \times s$ matrix of valid bits. As with the switch of

	Revsort	Columnsort, $\beta = 1/2$	Columnsort, $\beta = 5/8$	Columnsort, $\beta = 3/4$
pins per chip	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	$\Theta(n^{5/8})$	$\Theta(n^{3/4})$
chip count	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	$\Theta(n^{3/8})$	$\Theta(n^{1/4})$
load ratio	$1 - O(n^{3/4}/m)$	$1 - O(n/m)$	$1 - O(n^{3/4}/m)$	$1 - O(n^{1/4}/m)$
gate delays	$3 \lg n + O(1)$	$2 \lg n + O(1)$	$\frac{5}{2} \lg n + O(1)$	$3 \lg n + O(1)$
volume	$\Theta(n^{3/2})$	$\Theta(n^{3/2})$	$\Theta(n^{13/8})$	$\Theta(n^{7/4})$

Table 1: Resource measures for the Revsort-based partial concentrator switch and the values of β at which the Columnsort-based switch matches them asymptotically.

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \\ 15 & 16 & 17 \end{bmatrix} \qquad \begin{bmatrix} 0 & 6 & 12 \\ 1 & 7 & 13 \\ 2 & 8 & 14 \\ 3 & 9 & 15 \\ 4 & 10 & 16 \\ 5 & 11 & 17 \end{bmatrix}$$

row-major column-major

Figure 5: Row-major and column-major positions of elements in a 6×3 matrix.

the previous section, the m output wires of the switch correspond to the first m matrix entries.

We may identify a matrix entry by either its row and column position or by its position in row-major or column-major order. All numbering starts at 0. Thus, the rows are numbered $0, 1, \ldots, r-1$ and the columns are numbered $0, 1, \ldots, s-1$. The row-major position of the matrix entry in row i and column j is $RM(i,j) = si + j$, and its column-major position is $CM(i,j) = rj + i$. For example, Figure 5 shows the row-major and column-major positions of a 6×3 matrix. We have that $0 \leq RM(i,j), CM(i,j) < n$. The row and column position corresponding to the entry in row-major position x is $RM^{-1}(x) = (\lfloor x/s \rfloor, x \bmod s)$.

The partial concentrator switch is built from two stages, each stage containing s hyperconcentrator chips. Since the hyperconcentrator chips are combinational, so is the partial concentrator switch. Each r-by-r hyperconcentrator chip corresponds to a column of the underlying matrix, fully sorting the column. We shall denote by $H_{l,j}$ the jth hyperconcentrator chip in stage l, for $l = 1, 2$ and $0 \leq j < s$, with input wires $X_{l,j,0}, X_{l,j,1}, \ldots, X_{l,j,r-1}$ and output wires $Y_{l,j,0}, Y_{l,j,1}, \ldots, Y_{l,j,r-1}$. Wires $X_{l,j,i}$ and $Y_{l,j,i}$ correspond to the matrix element in row i and column j.

The wiring between stages 1 and 2 corresponds to converting the matrix from column-major to row-major ordering, using the composition of functions $RM^{-1} \circ CM$. We connect the output wire $Y_{1,j,i}$ to the input wire $X_{2,(rj+i) \bmod s, \lfloor (rj+i)/s \rfloor}$, for $0 \leq i < r$ and $0 \leq j < s$.

Once again, the output wires of the partial concentrator switch are the first m output wires of the matrix in row-major order. We use wires $Y_{2,j,i}$ for $0 \leq i < \lfloor m/s \rfloor$ and $0 \leq j < s$ or $i = \lfloor m/s \rfloor$ and $0 \leq j < m \bmod s$.

To show that this circuit $(s-1)^2$-nearsorts the valid bits, we first observe that its operation is equivalent to the following algorithm, which corresponds to the first three steps of Columnsort:

Algorithm 2 Given an $r \times s$ matrix of n elements, where $n = rs$, and matrix values of 0 or 1, perform the following three steps:

1. Fully sort the columns.
2. Convert the matrix from column-major to row-major order, i.e., move the element in row i and column j to row $\lfloor (rj+i)/s \rfloor$ and column $(rj + i) \bmod s$.
3. Fully sort the columns.

The two stages of hyperconcentrator chips correspond to steps 1 and 3, and the wiring between the stages corresponds to step 2. This correspondence between the circuit and Columnsort allows us to prove the following theorem.

Theorem 4 *The Columnsort-based construction yields an $(n, m, 1 - (s-1)^2/m)$ partial concentrator switch.*

Proof Leighton shows in [3] that Algorithm 2 is an $(s-1)^2$-nearsorter when the matrix elements are taken in row-major order. By Lemma 2, the circuit is an $(n, m, 1-(s-1)^2/m)$ partial concentrator switch when the outputs are taken in row-major order. □

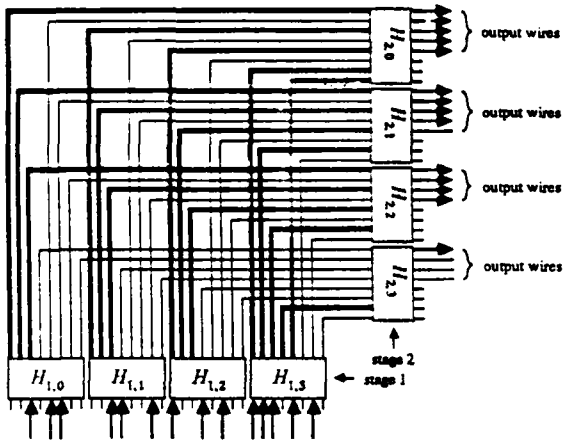


Figure 6: A two-dimensional layout of the Columnsort-based partial concentrator switch with $n = 32$ inputs and $m = 18$ outputs. The underlying matrix is 8×4. The electrical paths established by 14 valid messages are shown with heavy lines. The output wires are the first five output wires of hyperconcentrator chips $H_{2,0}$ and $H_{2,1}$ and the first four output wires of hyperconcentrator chips $H_{2,2}$ and $H_{2,3}$.

To achieve the results stated at the beginning of this section, we let $r = \Theta(n^\beta)$ and $s = \Theta(n^{1-\beta})$. To ensure that $n = rs$ and that s divides r as n increases, we require that we have $1/2 \leq \beta \leq 1$. The load ratio is then

$$\begin{aligned} \alpha &= 1 - \frac{(s-1)^2}{m} \\ &= 1 - \Theta\left(\frac{n^{2-2\beta}}{m}\right) . \end{aligned}$$

The number of chips is $2s = \Theta(n^{1-\beta})$, and each chip requires $2r = \Theta(n^\beta)$ data pins.

The delay through the switch is $2 \cdot 2\lg r + O(1) = 4\lg r + O(1)$. Letting $r \leq cn^\beta + o(n^\beta)$ for some constant c, we have that the delay is

$$\begin{aligned} 4\lg r + O(1) &\leq 4\lg(cn^\beta + o(n^\beta)) + O(1) \\ &\leq 4\lg((c+1)n^\beta) \quad \text{(for suff. large } n) \\ &= 4\beta\lg n + 4\beta\lg(c+1) \\ &= 4\beta\lg n + O(1) . \end{aligned}$$

A two-dimensional layout using $O(n^2)$ area is shown in Figure 6. As in the Revsort-based switch, we use $n \times n$ crossbar wiring to connect the stages.

Figure 7 shows a three-dimensional packaging of the switch using volume $\Theta(r^2 s) = \Theta(n^{1+\beta})$. As in Figure 6, we have $r = 8$ and $s = 4$. There are two stacks of boards, with each stack containing s boards

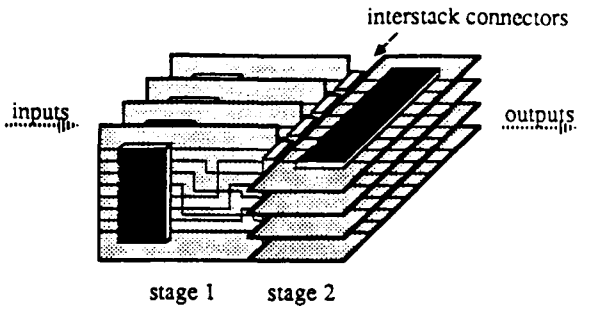


Figure 7: The three-dimensional packaging of the Columnsort-based partial concentrator switch for $r = 8$ and $s = 4$. Each stack contains s chips, each of which is an r-by-r hyperconcentrator. The wiring between the stages of chips performs the $RM^{-1} \circ CM$ permutation. The interstack connectors transpose the wires from vertical to horizontal alignment.

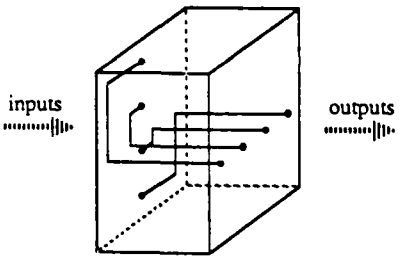


Figure 8: The transposition of w wires from vertical to horizontal alignment, shown for $w = 4$, using volume $\Theta(w^2)$.

and corresponding to one stage of hyperconcentrator chips, and each board containing one r-by-r hyperconcentrator chip.

The tricky part of this construction is the wiring between stages, which must perform the permutation $RM^{-1} \circ CM$. On the first stack, we group together output wires whose column-major numberings are congruent modulo s, or equivalently, those whose row numbers are congruent modulo s. Each such group contains r/s wires. In Figure 7, for example, since we have $s = 4$, we group together wires $H_{1,0,0}$ and $H_{1,0,4}$, $H_{1,0,1}$ and $H_{1,0,5}$, $H_{1,0,2}$ and $H_{1,0,6}$, $H_{1,0,3}$ and $H_{1,0,7}$, etc. In order to allow them to enter the stage 2 chips, these wires are then "transposed" in small interstack connectors to align them horizontally instead of vertically. Figure 8 shows one way to transpose a group of r/s wires in volume $\Theta((r/s)^2)$.

The first stack dominates the volume of this construction. We have s boards, and each board contains a $\Theta(r^2)$-area hyperconcentrator chip and an $O(r^2)$-

area wiring permutation. The total volume of each stack is thus $\Theta(r^2 s) = \Theta(n^{1+\beta})$. There are s^2 interstack connectors, each with volume $O((r/s)^2)$, for a total interstack volume of $O(r^2) = O(n^{2\beta})$. Since we have $\beta \leq 1$, the total interstack volume is $O(n^{1+\beta})$. The total volume of the partial concentrator switch is thus $\Theta(n^{1+\beta})$.

For both the two-dimensional and three-dimensional layouts, letting p, the number of pins per chip, be $\Theta(r)$, we use only $\Theta(s) = \Theta(n/p)$ chips. The three-dimensional layout, however, uses $s^2 = \Theta((n/p)^2)$ interstack connectors, but these connectors contain only wiring and no active components.

6 Concluding Remarks

In this section, we briefly discuss the characteristics of the partial concentrator switches we have seen and then discuss multichip hyperconcentrator switches. Finally, we pose some open questions.

Both of the partial concentrator switches we have examined are efficient in that they are relatively fast and can be packaged with a relatively low volume. They also allow air to flow through in all three dimensions and may thus be air-cooled.

The β parameter of the Columnsort-based switch defines a tradeoff continuum for the characteristics of the switch. As evidenced by Table 1, as the value of β increases, so do the number of pins per chip, delay, and volume, but the load ratio improves and the number of chips decreases.

Rather than simulating just the first steps of Revsort and Columnsort, one could simulate the full algorithms to fully sort the valid bits and thus build multichip hyperconcentrator switches. Compared to the partial concentrator switches presented above, such hyperconcentrator switches have increased delay, and a Revsort-based hyperconcentrator switch has a greater chip count and asymptotic volume than its partial concentrator counterpart.

Schnorr and Shamir show in [7] that if steps 1–3 of Algorithm 1 are repeated $\lceil \lg \lg \sqrt{n} \rceil$ times, the resulting matrix contains at most eight dirty rows. We can then complete the full sorting by running three iterations of the Shearsort algorithm [6]. An n-by-n hyperconcentrator switch based on the full Revsort algorithm consists of $\lceil \lg \lg \sqrt{n} \rceil$ repetitions of stacks 1 and 2 of Figure 4 followed by three pairs of different stacks that simulate Shearsort. (Each Shearsort stack consists of $\sqrt{n}$ boards, each of which contains a $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip and fixed permutation wiring.) A signal passes through $2 \lg \lg n + 4$ hyperconcentrator chips in such an n-by-n hyperconcentrator switch, incurring $4 \lg n \lg \lg n + 8 \lg n + O(\lg \lg n)$ gate delays. The switch uses a total of $\Theta(\sqrt{n} \lg \lg n)$ chips in volume $\Theta(n^{3/2} \lg \lg n)$.

Similarly, by simulating all eight steps of Columnsort, we can build a hyperconcentrator switch with the same asymptotic volume and chip count as the partial concentrator switch of Section 5. A signal passes through four chips and incurs $8\beta \lg n + O(1)$ gate delays through such an n-by-n hyperconcentrator switch.

Rather than wondering how fast a multichip hyperconcentrator switch we can build, we might ask for what functions $f(p)$ can we build an $(\Omega(f(p)), m, 1 - o(p/m))$ partial concentrator switch, given chips with p pins and using only two stages of chips. The Columnsort-based construction, for example, gives us $f(p) = p^{2-\epsilon}$ for any $0 < \epsilon \leq 1$. Can we achieve $f(p) = \Omega(p^2)$? In general, how large a function $f(p)$ can we achieve with k stages?

There may be ϵ-nearsorters based on networks other than the two-dimensional mesh to which we can apply Lemma 2. What types of partial concentrator switches can we build by applying Lemma 2 to other ϵ-nearsorters?

Acknowledgements

Thanks to Charles Leiserson for suggesting this line of research and for his frequent help and guidance. Thanks also to James Park for his helpful comments.

References

[1] T. H. Cormen, "Concentrator switches for routing messages in parallel computers," Masters thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., (1986), 66pp.

[2] T. H. Cormen and C. E. Leiserson, "A hyperconcentrator switch for routing bit-serial messages," *Proceedings of the 15th Annual International Conference on Parallel Processing*, (Aug. 1986), pp. 721–728.

[3] F. T. Leighton, "Tight bounds on the complexity of parallel sorting," *IEEE Transactions on Computers*, Vol. C-34, No. 4, (Apr. 1985), pp. 344–354.

[4] M. S. Pinsker, "On the complexity of a concentrator," *Proceedings of the 7th International Teletraffic Conference*, Stockholm, (1973), pp. 318/1–318/4.

[5] N. Pippenger, "Superconcentrators," *SIAM Journal on Computing*, Vol. 6, No. 2. (June 1977), pp. 298–304.

[6] I. D. Scherson, S. Sen, and A. Shamir, "Shear sort: a true two-dimensional sorting technique for VLSI networks," *Proceedings of the 15th Annual International Conference on Parallel Processing*, (Aug. 1986), pp. 903–908.

[7] C. P. Schnorr and A. Shamir, "An optimal sorting algorithm for mesh connected computers," *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, (May 1986), pp. 255–263.

[8] L. G. Valiant, "Graph-theoretic properties in computational complexity," *JCSS*, Vol. 13, No. 3, (Dec. 1976), pp. 278–285.

EFFICIENT GRAPH ALGORITHMS
FOR
SEQUENTIAL AND PARALLEL COMPUTERS

by

Andrew Vladislav Goldberg

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

In this thesis we study graph algorithms, both in sequential and parallel contexts. In the following outline of the thesis, algorithm complexities are stated in terms of the number of vertices n, the number of edges m, the largest absolute value of capacities U, and the largest absolute value of costs C.

In Chapter 1 we introduce a new approach to the maximum flow problem that leads to better algorithms for the problem. These algorithms include an $O(nm\log(n^2/m))$ time sequential algorithm, an $O(n^2\log n)$ time parallel algorithm that uses $O(n)$ processors and $O(m)$ memory, and both synchronous and asynchronous distributed algorithms.

Chapter 2 is devoted to the minimum cost flow problem, which is a generalization of the maximum flow problem. We introduce a framework that allows the generalization of the maximum flow techniques to the minimum-cost flow problem. This framework allows us to design efficient algorithms for the minimum-cost flow problem. We exhibit $O(nm\log(n)\log(nC))$, $O(n^{5/3}m^{2/3}\log(nC))$, and $O(n^3\log(nC))$ time sequential algorithms as well as parallel and distributed algorithms.

In Chapter 3 we address implementation of parallel algorithms through a case-study of an implementation of a parallel maximum flow algorithm. Parallel prefix operations play an important role in our implementation. We present experimental results achieved by the implementation.

Parallel symmetry-breaking techniques are the main topic of Chapter 4. We give an $O(\lg^* n)$ algorithm for 3-coloring a rooted tree. This algorithm is used to improve several parallel algorithms, including algorithms for $\Delta+1$-coloring and finding maximal independent set in constant-degree graphs, 5-coloring planar graphs, and finding a maximal matching in planar graphs. We also prove lower bounds on the parallel complexity of the maximal independent set problem and the problem of 2-coloring a rooted tree.

Thesis Supervisor: Professor Charles E. Leiserson

Title: Associate Professor of Computer Science and Engineering

Nonlinear Dynamic Maximum Power Theorem

John L. Wyatt, Jr.

ABSTRACT

This paper considers the problem of maximizing the energy or average power transfer from a nonlinear dynamic source. The main theorem includes as special cases the standard linear result $Y_{load} = Y^*_{source}$ and a recent finding for nonlinear resistive networks. An operator equation for the optimal output voltage $\hat{v}(\cdot)$ is derived, and a numerical method for solving it is given.

This research was supported by the National Science Foundation under Grant No. ECS 8310941 and the Defense Advanced Research Projects Agency under Contract No. N00014-80-C-0622.

Earlier versions of this work appeared in [1], [2].

The author is with the Research Laboratory of Electronics and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.